AN EXPERT SYSTEM APPROACH TO ANALOG CIRCUIT SYNTHESIS

A Thesis Submitted

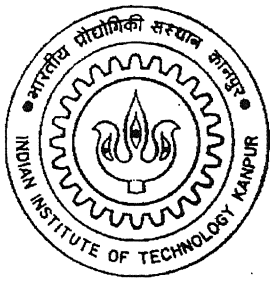
in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Venigalla Ravindra Babu



to the

DEPARTMENT OF
ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

March 1996.

1 7 MAY 1996
LIBRARY
KANPUR
Acc. No. A. 121549

EE-1996-M-BAB-EXP

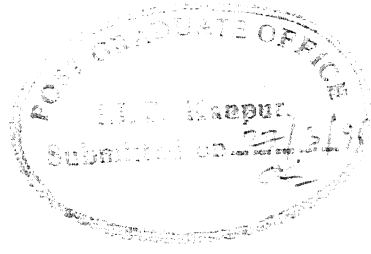
CERTIFICATE

It is certified that the thesis entitled *"AN EXPERT SYSTEM APPROACH TO ANALOG CIRCUIT SYNTHESIS"* by **VENIGALLA RAVINDRA BABU** has been carried out under our supervision and that it has not been submitted elsewhere for the award of a degree.

(**Dr. B. Mazhari**)
Asst. Professor
Dept. of Electrical Engg.
IIT Kanpur, INDIA – 208 016

(**Dr. M. M. Hasan**)
AICTE Emeritus Fellow
Dept. of Electrical Engg.
IIT Kanpur, INDIA – 208 016

March 1996.



Dedicated to

Dr. K. S. R. Krishna Prasad

Professor, Dept. of ECE

REC Warangal, INDIA – 506 004.

who helped me to come this far

Acknowledgements

I express my deepest sense of gratitude and appreciation to my thesis supervisor **Dr. B. Mazhari** for his guidance, encouragement and the freedom which he gave is responsible for the piece of work. His ideas, at times when this work went into troubles, have enabled me to successfully complete this work. His criticisms have improved the clarity of presentation of this thesis. I express my sincere gratitude to my co-supervisor **Dr. M. M. Hasan** for spending his valuable time during the work. I thank them for giving the problem, which is in my area of interest.

I am greatful to Dr. A. K. Dutta for the contacts I have had with him during the course work, which developed in me the skills required for carrying out the present work. I thank my teachers Dr. R. Sharan, Dr. S. Kar, Dr. M. B. Patil and Dr. S. K. Roy who have taught me various courses during my academic programme. I thank Mr. Aditya G. Kiran, Mr. D. Dileep Kumar and Mr. K.Srinivas for their help in programming.

It is the time to recall the moments spent in the wings, lawns, and Hall III canteen with *Basha and Reddy*. I can never forget the love and affection received from them during my 20 months stay here. They gave an ideal company for relaxing and breaking the regular routine. It is a pleasure to thank my other friends and classmates specially *Deshpande, Prashant* and *Sandeep* who made the stay a memorable one.

My special thanks to the authorities of IIT Kanpur who make this institution an excellent learning place.

V. R. Babu

Abstract

An analog design methodology that uses the assistance of an expert system is presented. A top–down analytical knowledge based design that fully exploits any inherent hierarchy in the system, is first generated. Exploiting hierarchy permits the design process to be recast as a sequence of smaller design tasks, and permits the sub–blocks to be reused in different contexts. The initial design is further refined by making use of an expert system that learns rules as it gains experience. The design methodology is demonstrated through the example of a CMOS operational amplifier.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

The growing complexity of the VLSI circuits and the necessity of reducing the design cost of custom Integrated Circuits forced the development of a large number of Computer Aided Design (CAD) tools, which improve the design process in several ways:

1. By shortening design times.

2. By simplifying design process: As a consequence more designers, from novice designers to system designers or expert designers, will be able to design standard circuits.

3. By improving the likelihood of error-free designs: Automating error prone design tasks reduces the probability of making errors and therefore, decreases the design cycles/success ratio.

4. By retaining expert design knowledge: The knowledge held by the design systems can be used over and over again to design circuits. In addition, it can be conveyed to novice users in the form of design examples, explanations of behavior, suggestions for modifications or conclusions.

While sophisticated CAD techniques have been developed to automate the design of digital circuitry, the process towards automating the analog circuit design has been relatively slow. In part, this is due to the highly complex and knowledge–intensive

nature [1] of the analog design. However, with increasing demand for single chip mixed analog/digital VLSI IC's, the need for Analog Design Automation (ADA) tools has become really pressing. Although analog circuits occupy a small fraction of these chips, their design has become a critical bottleneck in the overall design process because they are still 'hand–crafted' by expert circuit designers.

Till 1985, very few ADA systems were reported. Many of these systems provided only assistance to the designer and showed strong influence from the digital domain. In the last decade, however, a plethora of prototype systems, many of which are capable of handling full–custom designs, have been reported. The various design automated approaches are summarized in Fig. 1.1.

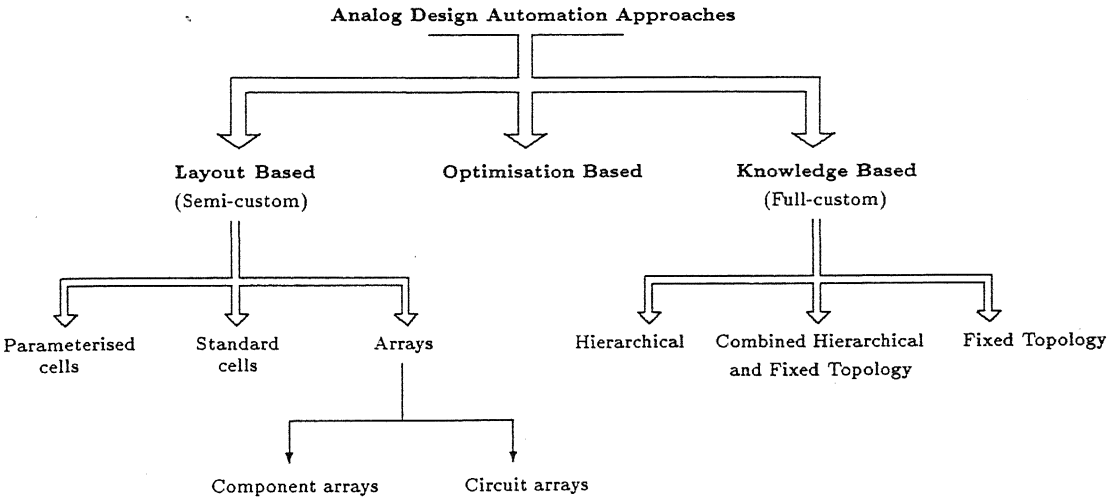


Figure 1.1: Classification of ADA

The first attempts towards ADA were optimization–based approach. Systems such as DELIGHT.SPICE [2] and the recent ADOPT [3] consider the sizing of transistors of a user–given circuit topology as an optimization problem. Typically these systems employ optimization algorithms to iteratively adjust transistor sizes in order to meet user input constraints and objectives. A simulator is used within the optimization loop to assess the performance of the circuit at each iteration. Systems based on numerical optimization techniques are independent of the actual circuit, technology and the fabrication process used. However use of such systems has been limited because the circuit

designer has to be familiar with optimization. If a good initial design is not supplied, then the optimization routine may converge to some local minimum corresponding to a faulty design. Further, these systems are slow, since they involve circuit simulation within the optimization loop.

Layout–based design approach is an adaptation of approaches such as standard cell, gate–array and parameterized cell found in digital design automation. It is also referred to as the semi–custom bottom–up approach, because designs are controlled to a large extent by the layout. Analog arrays, varying from single component arrays to circuit arrays are, pre–designed. The required functions are designed by appropriately programming one or more levels of interconnect. Typical examples of circuit arrays are SCA–6 and SCA–12 Switched Capacitor filter arrays with the assorted tools like SLIDE [4]. Designing with analog arrays has the drawback that they do not provide the necessary design flexibility required for high performance analog circuits, thereby restricting the designer to a limited range of available active and passive components and their values. They can realize only a small number of discrete points from a wide and continuous performance space. Further they are not cost–effective in terms of silicon usage, since any unused components or circuits in the array simply waste silicon area. Standard cells [5], [6] address this latter problem of silicon usage. They are pre–designed and layed–out blocks of varying complexity that reside in the database of the ADA system. The required function is implemented by assembling the necessary cells and then placing and routing. However use of such systems has been limited because of the difficulty in configuring and maintaining a rich enough library of cells to accommodate a wide spectrum of possible specifications. Parameterized cells are similar to standard cells, but with some additional flexibility gained by allowing some customization of the cells. The degree of flexibility provided is directly dependent on the sophistication of the respective module generator. In this way, components can now have continuous values. With the application of this approach, AIDE2 [7] has demonstrated the design of several circuits.

proaches to ADA to circumvent the disadvantages of the previous methods. They address the design task in a full–custom way, thereby allowing for maximum flexibility and better coverage of the circuit's performance space. The prominent design approaches that have been used are hierarchical, fixed–topology and a combination of both. Hierarchical approach involves the breaking of the required circuit into smaller distinct parts or blocks. Each of these parts is assigned a set of specifications so that, if met, the combination of the performance of these parts will yield the desired circuit performance. The same procedure is repeated to lower levels. The number of hierarchical levels depends upon the complexity of the circuit. To carry out such partitioning of circuits and decomposition of specifications, knowledge is required which is stored in the form of design equations and heuristics. This approach has been realised by a number of systems, such as BLADES [9], OASYS [11], AN–COM [12], ANAGRAM [22] and ARIADNE [34]. Fixed–Topology approach differs widely from the first one. Systems following this approach employ a sizing method within a given fixed circuit topology. These fixed, unsized topologies are stored in a knowledge base, together with the necessary knowledge for dimensioning the devices. Significant problem with this approach is the large overhead that is created whenever a new topology is added. Some of the systems that follow this approach include IDAC [8] and OPASYN [13]. Combined Hierarchical and Fixed–Topology approach combines features of both the above described schemes, thereby providing an additional advantage of topology modification. This can lead to a smaller circuit library and a wider coverage of circuit performances. However, they are not as flexible as fully hierarchical systems. Some of the systems that follow this approach include CHIPAIDE [25] and CAMP [27]. Table 1.1 shows the comparision among various knowledge based systems.

Name	Developed by	Layout synthesis	Link to external simulator	Optimi-zation	Comme-rtial	Hiera-rchy	Worst-case design	Sizing method
ACACIA	Carnegie Melon Univ., Pittsburgh	✓	✓	✓		✓		Knowledge based synthesis, plus expert system
ADAM	CSEM, Switzerland	✓	✓	✓			✓	Hard-coded design plan, steepest descent optimization.
OPASYN	Univ of California Berkeley	✓	✓	✓				Heuristic circuit selection, steepest descent optimization.
MIDAS	Silicon and Software System	✓			✓	✓		Knowledge based synthesis via compilable language.
BLADES	AT&T		✓			✓		Knowledge based synthesis based on expert system.
ARIADNE	Katholieke Univ., Leuven	✓	✓	✓		✓	✓	Optimization based on analytic models drawn from symbolic analysis.
AN-COM	General Electric R&D Center, NY	✓				✓		Knowledge based synthesis.
CHIPAIDE	Imperical College London	✓	✓	✓		✓	✓	Hard-coded design plan.

Table 1.1: Comparision among various knowledge based systems.

Since automated circuit generation is often done with the help of approximate analytical equations, the generated circuit may fail to meet the desired performance specifications. In such cases, the design needs to be corrected. The correction usually involves tuning of component values, however, in some cases, it may even require topology modification. The approaches for circuit correction are as diverse as numerical optimization [2], [8], [13] to expert system qualitative reasoning [20], [9] - [11]. The expert system approach is especially attractive, since in real life situation, the tuning of circuit is often done by expert using a variety of heuristic rules.

In the present work a combined hierarchical and fixed topology approach is used for entire circuit synthesis and an expert system based optimization is subsequently used for circuit correction. The design methodology is described in detail in Chapter 2. The results obtained for our system are documented in Chapter 3. Chapter 4 concludes with some suggestions for future work.

CHAPTER 2

A NEW DESIGN SYSTEM - AN OVERVIEW

The block diagram of a knowledge based analog design system implemented in this work is shown in Fig. 2.1. It comprises of three modules: the *synthesizer*, the *optimizer* and the *task coordinator*. Each of these modules has well–defined task and moreover, the structure of the system enjoys high degree of modularity. The aim of the synthesizer is to transform a set of performance and process specifications into an appropriate–sized transistor topology. The initial design obtained from the synthesizer is further refined using an optimizer. The aim of the task coordinator is to establish effective communication between the synthesizer, simulator and the knowledge base. It also keeps track of various activities.

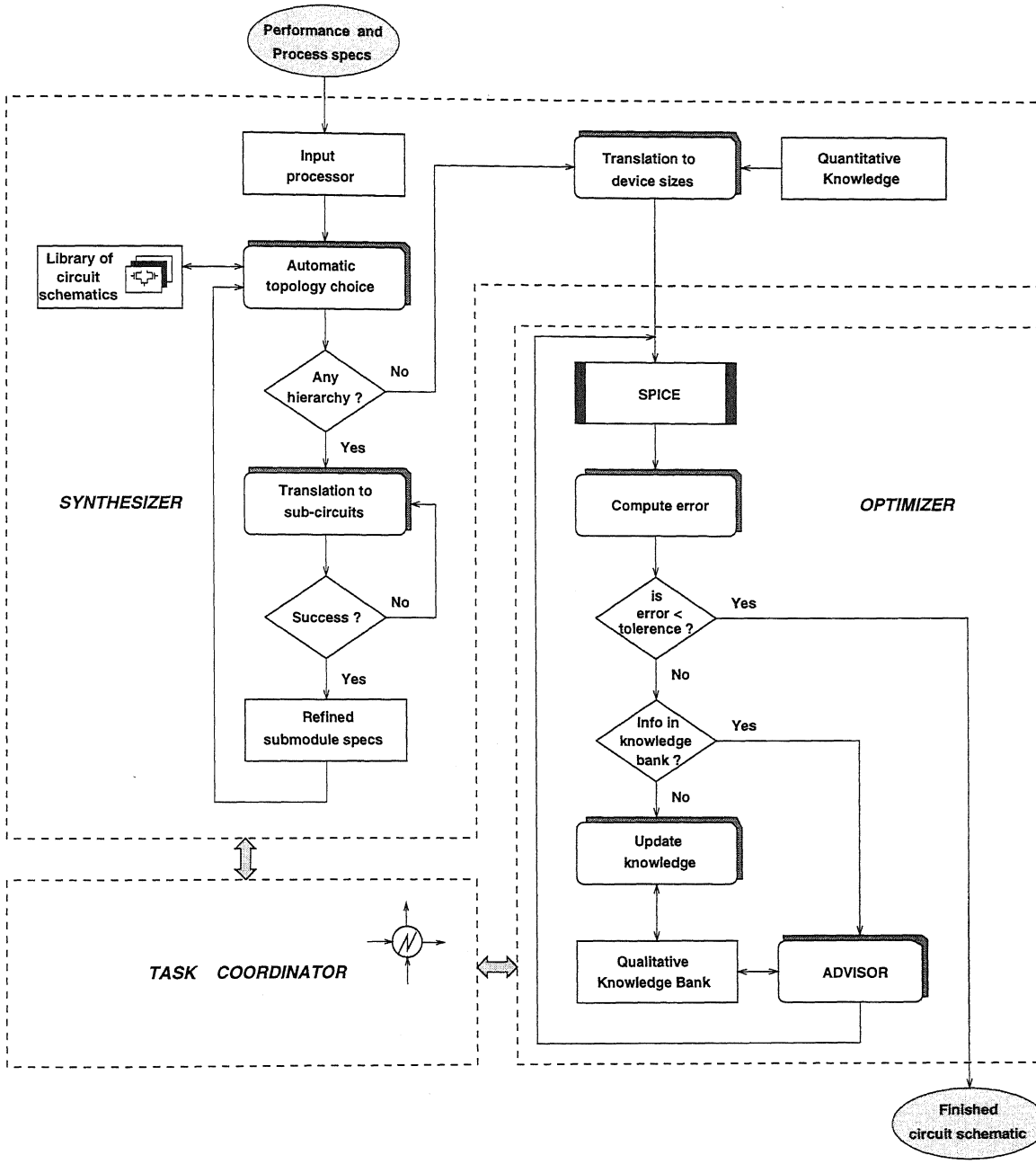


Figure 2.1: The Design Methodology.

2.1 The Synthesizer

The synthesizer uses formal knowledge to generate a circuit topology that would meet the given set of specifications. It exploits hierarchy by decomposing the specifications of the circuit at the top to specifications for its associated sub–circuit blocks. This process of decomposition is repeated to lower levels of hierarchy. Then, during the reverse process, the expected performance of the lower level blocks are combined together to synthesize blocks of higher levels. Hierarchical design methodology [9]-[12] is employed for a number of reasons:

1. Hierarchy allows the breaking down of a large and difficult problems into smaller and easier ones.

2. Hierarchical structuring of circuit blocks permits the reusability of design knowledge.

3. Hierarchy provides a measure of generality, in that the sub–blocks can be reused in different contexts.

4. Hierarchy covers wide designable performance range with a smaller number of circuit blocks, since a single circuit block can be constructed with different circuit topologies.

Fig. 2.2 shows the hierarchy in a successive–approximation type A/D converter. Hierarchical representation however has the disadvantage that knowledge made in one sub–module is not used while designing another sub–module. Many of the design tricks precisely exploit this knowledge to achieve highly efficient designs.

The synthesis consists of architecture selection and specification translation, each of which are described in detail below.

2.1.1 The Architecture Selection

The first step in synthesis is the selection of an architecture that could meet the user given performance specifications. An architecture (or a design style or a candidate style or a topology) is defined as the description of a circuit block at a lower level

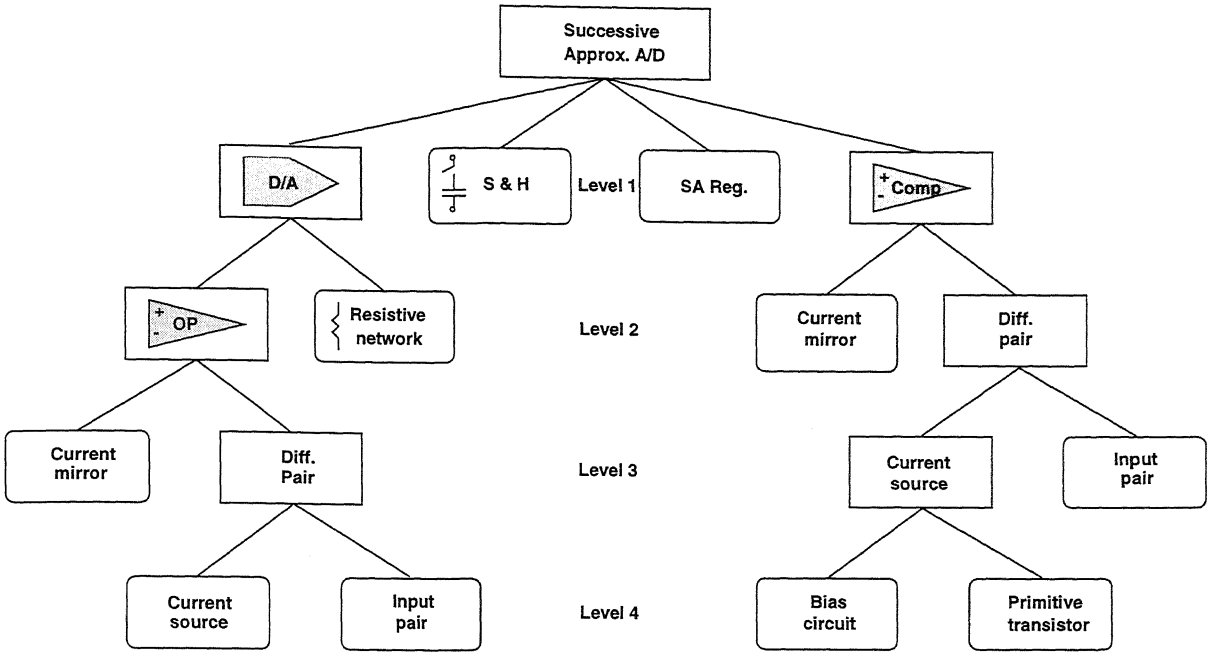


Figure 2.2: Example of hierarchy in analog circuits.

of hierarchical abstraction, in terms of its constituent lower level circuit block variety and interconnectivity. Fig. 2.3 shows an example of the architecture of a two–stage operational amplifier, illustrating schematically the required circuit blocks and how they are connected.

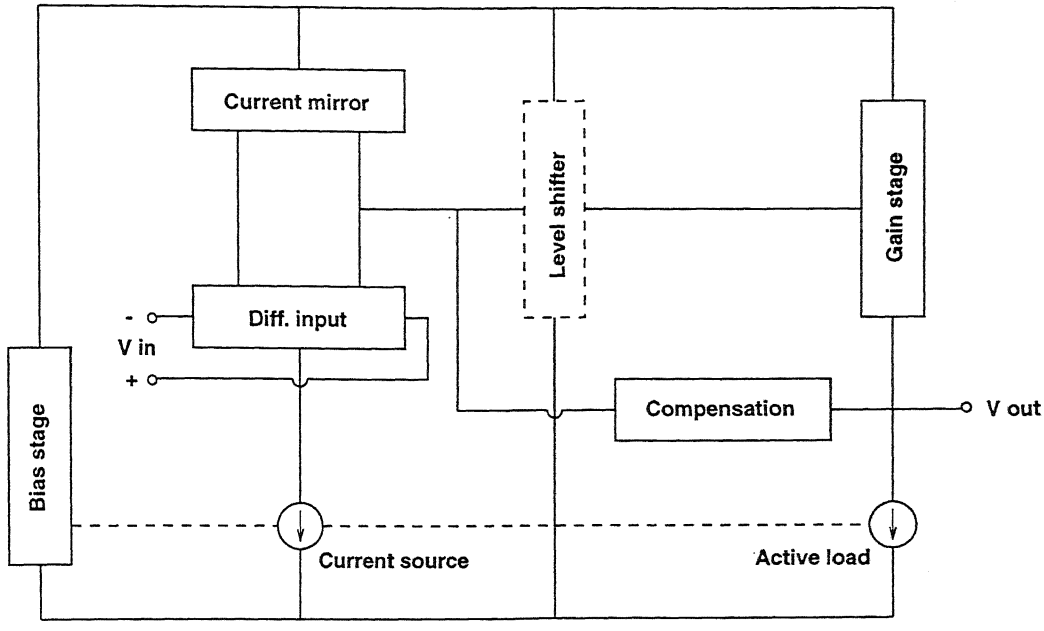


Figure 2.3: Two-stage operational amplifier architecture.

Each architecture corresponds only to one circuit block, whereas a circuit block can have more than one architecture. For example, the OP-AMP module has a number of possible architectures, such as a two-stage (shown in Fig. 2.3), single stage (complementary push-pull), folded-cascode, etc. Although each of these architectures may be fundamentally different from each other in terms of the lower level circuit block configurations, basic function of the module remains the same. An architecture may include two types of circuit blocks: fixed and conditional. Fixed circuit blocks appear in all implementations of that architecture, whereas conditional circuit blocks appear only if a predefined condition is satisfied. For example, the level shifting block in the OP-AMP architecture shown in Fig. 2.3, is necessary if the current mirror is implemented by cascode configuration. Even at the lowest level in the hierarchy, there are several architectures available to implement a particular function. For example, consider the three realizations of a transistor block as shown in Fig. 2.4. These different styles provide the same functional behavior but offer different performance tradeoffs.

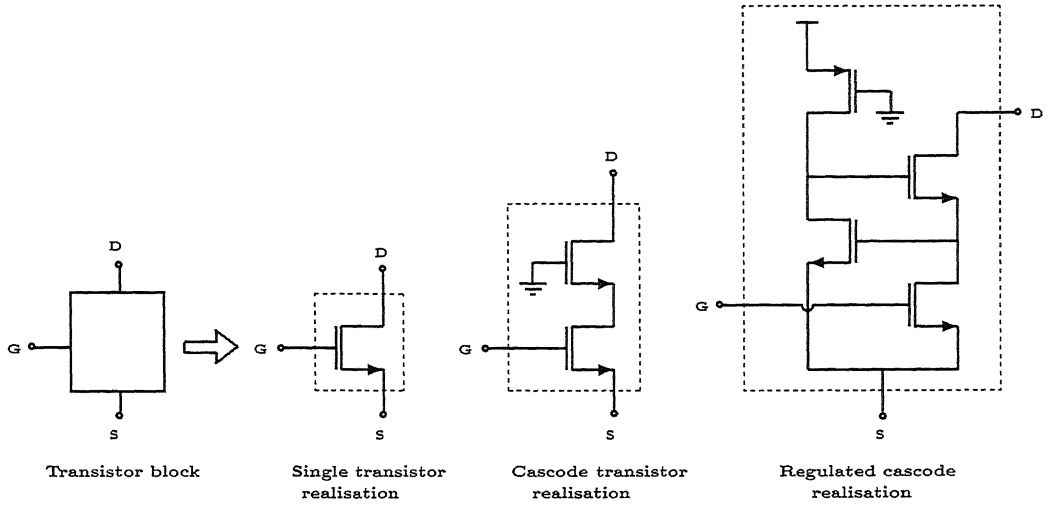


Figure 2.4: Transistor block realization.

The automatic topology choice: At each level of hierarchy, the different available architectures have been ranked according to various performance criteria so as to facilitate the selection of an architecture that is most likely to meet the specifications. For example in Fig. 2.4 the ordering for improving output resistance is: the single transistor, the cascode and then the regulated cascode. In contrast to that, the ordering for improving minimum output voltage is: the cascode, the regulated cascode and then the single transistor.

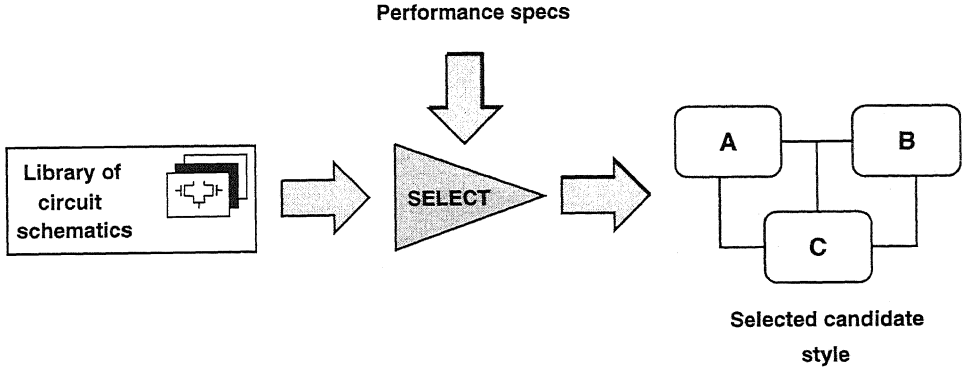


Figure 2.5: Example of architecture selection.

The architecture selection mechanism, as shown in Fig. 2.5 used in the present system is fast, simple and is good for designing lower–level circuit blocks where number of specifications are less. However, as we move to higher level blocks, this selection method may be inefficient.

2.1.2 The Translation

The translation, also known as decomposition or refinement, is the key to hierarchical design. It refers to breaking a set of performance specifications of a circuit into smaller circuit blocks associated with it at lower levels of hierarchy. This hierarchical decomposition is different for different analog circuits and it is a part of the formal knowledge for the design of that circuit, *i.e.*, it is derived from the expertise of the designer. We employed the most common and straight top–down decomposition. In this approach, the specifications of higher level circuit blocks are broken down into separate specifications of smaller level circuit blocks as shown in Fig. 2.6.

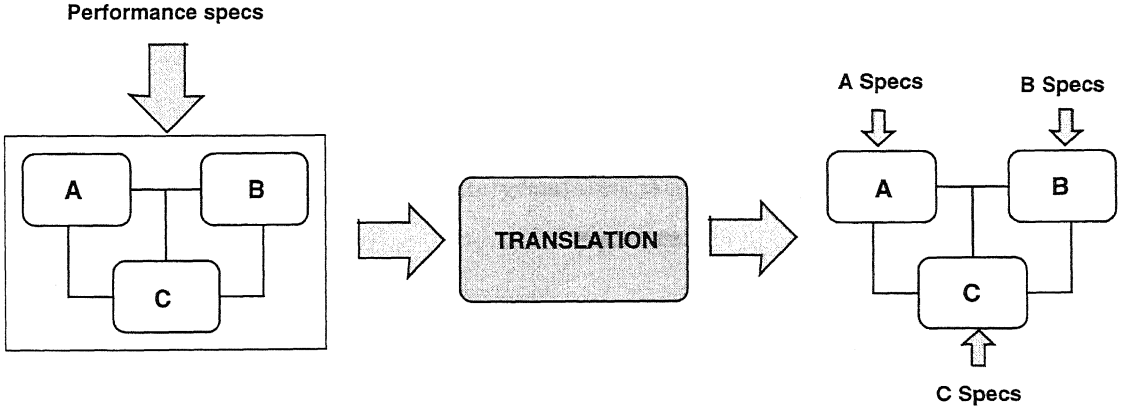


Figure 2.6: Example of translation of specifications.

This mechanism is most appropriate in cases where higher level specifications can be satisfied independently by a number of lower-level blocks and individual performances of lower-level blocks are also fairly decoupled. However, in many analog circuits, there is a tight coupling among circuit sub–blocks, making decomposition less straight forward. Consider for example, the simple two–stage OP–AMP shown in Fig. 2.3. This OP–AMP consists of fixed configurations of building blocks such as current mirrors, differential

pairs, bias stages, current sources, loads and gain stages. The input performance specifications for OP–AMP are dc gain, unity gain bandwidth (UGB), slew rate (SR), CMRR, phase margin (PM), power dissipation (PD), etc. The overall gain of the OP–AMP is a product of the gain due to the differential amplifier and that of the gain stage. The partitioning of the overall gain between these two stages is not straightforward, because performance specifications such as UGB, SR and phase margin also depend on this choice. As a result, a few iterations may need to be done to achieve the optimal partitioning.

The final stage of the decomposition process is the assignment of refined specifications to some primitive devices. This assignment will be carried out by the use of formal knowledge which for OP–AMP's can be found in [33]-[35]. These equations are less accurate being based on simplified device models than SPICE circuit models [23], [24], however, for initial design purposes they are sufficiently good approximations. This process has to be accurate and reliable, since it will determine, to a certain extent, the success of the designed circuit block.

2.2 The Optimizer

After the initial synthesis is done, the generated circuit is evaluated using SPICE. If the simulated performance is found to differ significantly, then the system enters into the optimization loop for circuit correction.

The various approaches towards optimization can be divided into either numerical [2], [8], [13] or rule–based [9] - [11], [20]. Numerical optimization is based on algorithms such as Newton–Raphson's for solution of non–linear equations. The first step in this approach is to obtain an incremental linear model of the circuit that relates specifications to the circuit parameters (transistor sizes in the present case). This incremental model is then inverted to generate a precise quantitative recommendation for correction of the circuit. Consider for example a two stage OP–AMP circuit shown in Fig. 2.3. The important performance specifications in this case are midband gain (A_{v0}), unity gain frequency (f_0), slew rate (SR), phase margin (PM), power dissipation (PD). These specifications depend on sizes of transistors M_1, M_2, $\cdots$ and M_n. The incremen-

tal model for this system is a set of linear equations

$$\begin{pmatrix} \Delta A_{vo} \\ \Delta f_0 \\ \vdots \\ \vdots \\ \Delta PD \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \cdots & a_{2n} \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ a_{m1} & a_{m2} & \cdots & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} \Delta(W/L)_1 \\ \Delta(W/L)_2 \\ \vdots \\ \vdots \\ \Delta(W/L)_n \end{pmatrix} \tag{2.1}$$

To determine the parameters of the model, a circuit simulator has to be used. For example, a_{11} can be obtained by changing (W/L) of transistor M_1 by 10% and determining the corresponding changes in A_{v0}, f_0, SR, PM and PD. In the present example, the circuit simulator would have to be used $5n$ times, where n represents the total number of circuit parameters. After the model is obtained, the linear equations are solved to determine the changes in (W/L)s needed to achieve the required corrections in A_{v0}, UGF, SR, PM and PD. Because of the use of the circuit simulator to generate the incremental model the optimization process gets slowed down.

In the rule–based expert system approach to optimization, the use of time consuming circuit simulator is eliminated. A qualitative rule–base model of the circuit is permanently stored in the design system. These heuristic rules obtained from circuit experts are of the form 'If $(W/L)_1$ increases then A_{v0} decreases', etc. This qualitative model of the system could also be generated through qualitative reasoning approach [26]. Using these rules, a qualitative recommendation for the circuit correction is generated. These qualitative recommendations are then translated into quantitative changes. The entire optimization process mimics the numerical Newton–Raphson's method as shown in Fig. 2.7.

Instead of precise numerical values for the slope, we have in this case just a qualitative 'increase', 'decrease' or 'no change'. This approach is simple, but because the simple qualitative model does not capture the full circuit complexity, large number of iterations may be needed for the optimization process, especially if the design is far from the desired one. Further optimization in some cases can be poor. For example, suppose one specification improves by changing a circuit parameter while another degrades with it. In this case it seems that only a tradeoff is possible. However, it is likely that one specification improves by a large amount while another degrades only

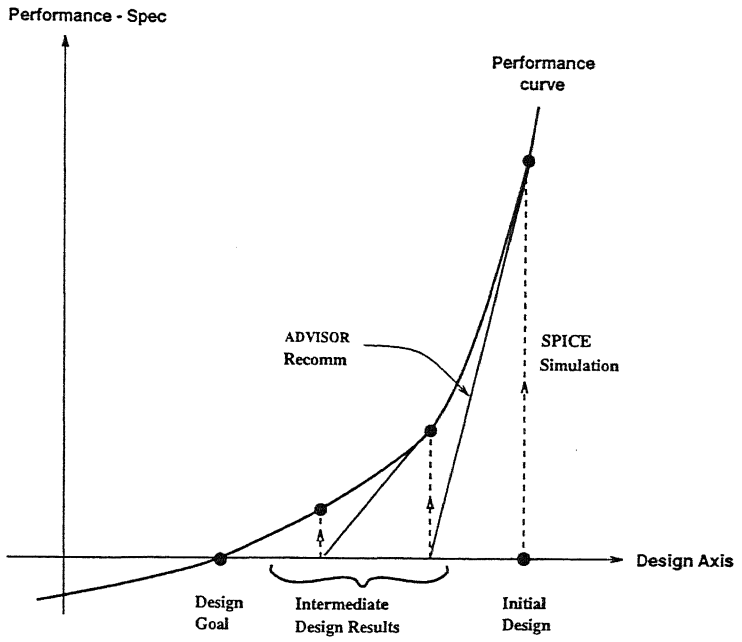


Figure 2.7: Converging nature of design process.

by a little. It is obvious that if this knowledge is available, better optimization could be achieved.

	M0	M1	M3	M7	M8	M9	Cc
Ao	s	N	N	n	n	N	n
UGF	N	A	N	N	N	a	l
SR	L	N	N	N	n	N	a
PM	N	N	n	S	S	a	S
PD	A	N	s	S	S	N	N

	M0	M1	M3	M7	M8	M9	Cc
Ao	↓	–	–	–	–	–	–
UGF	–	↑	–	–	–	↓	↓
SR	↑	–	–	–	–	–	↓
PM	–	–	–	↑	↑	↓	↑
PD	↑	–	↓	↑	↑	–	–

n : very small negative	N : very small positive	↓ : negative
s : small negative	S : small positive	↑ : positive
a : average negative	A : average positive	– : no change
l : large negative	L : large positive	
x : very large negative	N : very large positive	

Table 2.1: Comparison between five level and two level quantizations.

Consider for example a two level and a five level qualitative model of an OP–AMP for a particular design subspace as shown in Table 2.1. Suppose that the circuit correction requires both A_{v0} as well as SR to be increased. The two level model says that a change in M_0 will not be able to force the circuit closer to the desired specifications. However, the five level model shows that SR changes by a large amount, while A_{v0} changes only a little. Hence, there is scope for further optimization. From these considerations, it appears that if the qualitative dependencies are quantized into finer levels such as 'large increase', 'medium increase' and 'small increase', better optimization could be achieved and number of iterations may also be less.

This approach adopted in the present system, is intermediate between the numerical and simple rule–based system (Fig. 2.8).

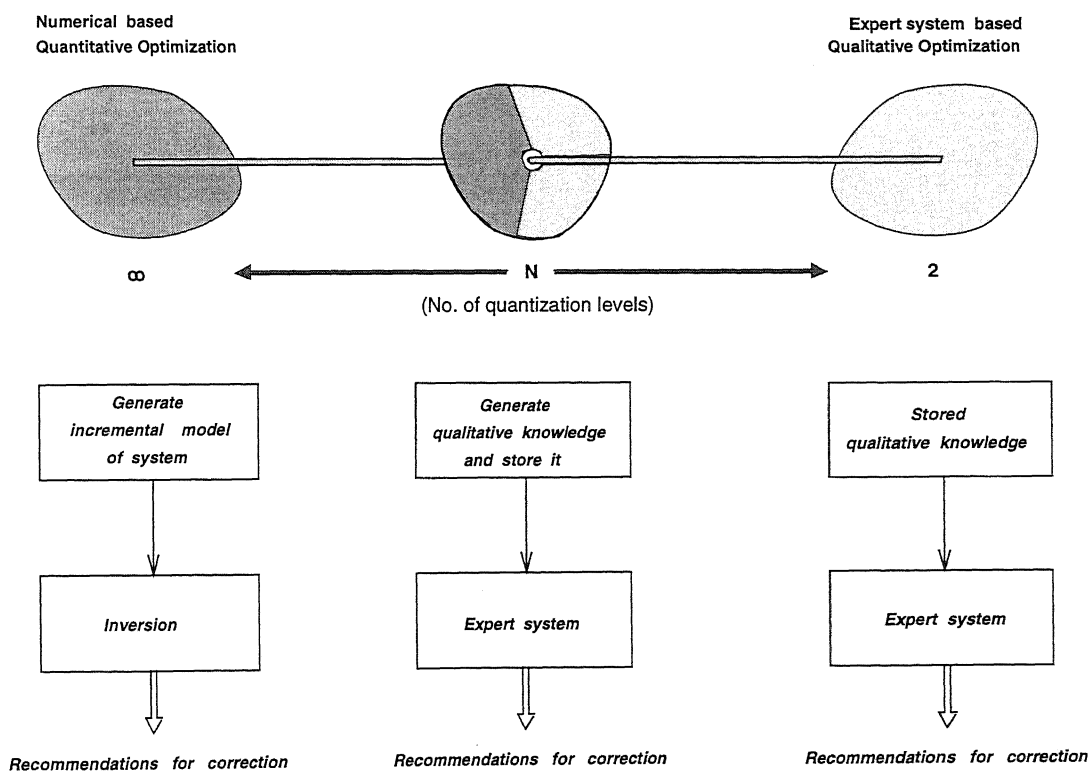
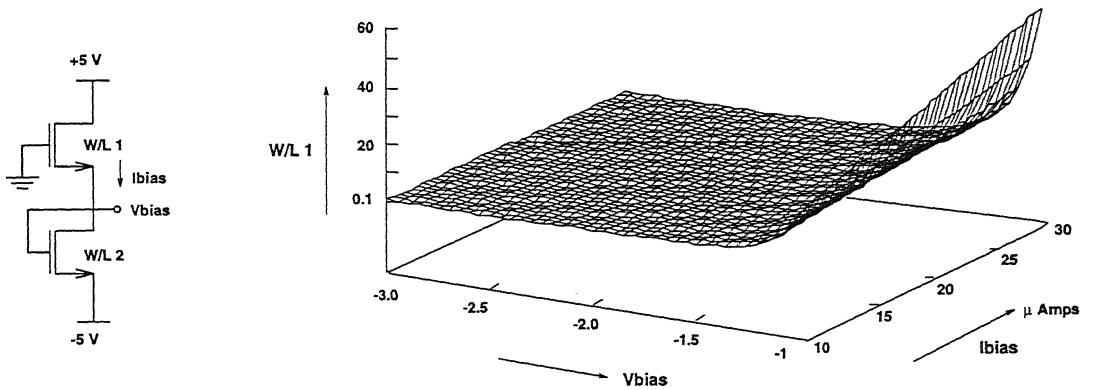


Figure 2.8: Different approaches towards optimization.

The numerical approach represents infinite quantization of the dependencies of specifications on circuit parameters, while the traditional rule–based systems represents a

two level quantization. For a two or three level quantization of qualitative dependencies, the rules may be known, but for finer quantizations, they have to be generated. Further, these rules will be different for different combinations of performance specifications, since analog circuits are often quite nonlinear. Hence the performance space needs to be discretized. Consider for example a simple bias stage at process specifications threshold voltage $V_{TN} = 0.8$ volts, transconductance parameter $K_p = 25\ \mu Amp/volt^2$ and channel length modulation parameter $\lambda = 0.03\ volt^{-1}$ as shown in Fig. 2.9.



(a) (b)

Figure 2.9: (a) A simple 2-transistor bias stage, and
(b) Variation of device size with respect to performance specs.

It is clear from Fig.2.9(b) that for a particular value of I_{bias}, the variation of V_{bias} with respect to $(W/L)_1$ is small in the range $V_{bias} = -3.0$ to -1.5 volts, while variation is sharp in the other region. In this case it becomes natural to discretize the V_{bias} axis into two regions at $V_{bias} = -1.5$ volts.

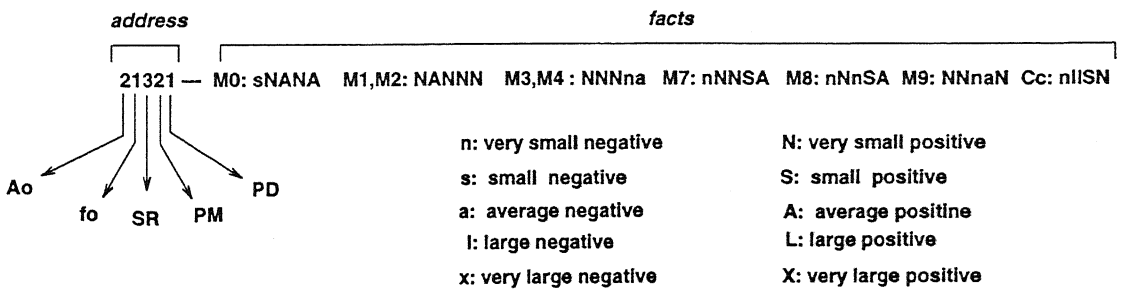
The aim of discretization is therefore to cut performance space into such performance subspaces where the circuit behavior is qualitatively different. However, the difficulty is that the right division of performance space may not be known beforehand. In our approach, we first divide the performance space into an arbitrarily large number of subspaces, within each of which the circuit behavior is expected to be uniform. The qualitative model of the circuit is not generated for each of these subspaces beforehand, but only when it is required in the optimization process. However, once the model is

generated, it is stored in the system and can be reused subsequently. The qualitative model is a qualitative abstraction of a quantitative model generated using a circuit simulator. Our approach uses the circuit simulator in the optimization loop only if qualitative knowledge is not available. With time this becomes less frequent and the system gets faster. In a way this mimics the learning mechanism of an expert, who gains knowledge everytime he designs a circuit. Basically, our optimization consists of an expert system and its associated knowledge base (Fig. 2.10).

2.2.1 The Knowledge Base

The knowledge base contains dependencies of performance specifications on circuit parameters (device sizes) for different subspaces for all types of topologies. Each subspace is specified by an *address* which is a string containing 'n' number of digits, each of which can take 'm' number of values, where n is total number of specifications and m is the number of levels into which a specification has been quantized. Ex: Bias circuit shown in Fig. 2.9(a) can take the address of 11 or 21. Here the first digit represents V_{bias} while the second digit represents I_{bias}. V_{bias} is discretized into two levels *i.e.*, -3.0 to -1.5 and -1.5 to -1.0 whereas I_{bias} is quantized to one level only, since it is linear throughout the range.

All the dependencies in the knowledge base are in the form of *facts*. A fact is a relation between objects. Ex: I_{bias} increases with (W/L). This fact consists of two objects, called "I_{bias}" and "(W/L)", and a relationship, called "increases". We represent the knowledge in the form of:



The interpretation is that, if (W/L) of M0 increases, gain A_o decreases by small amount, no change in unity gain frequency f_o, slew rate SR increases by average

amount, no change in phase margin PM and power dissipation PD increases by average amount and so on.

2.2.2 The Expert System

The aim of the expert system is to generate recommendations for circuit correction. It checks if knowledge for correction of the circuit in its present state exists or not. If knowledge is found, then the expert system generates recommendations (Part of expert system implementation is given in Appendix B).

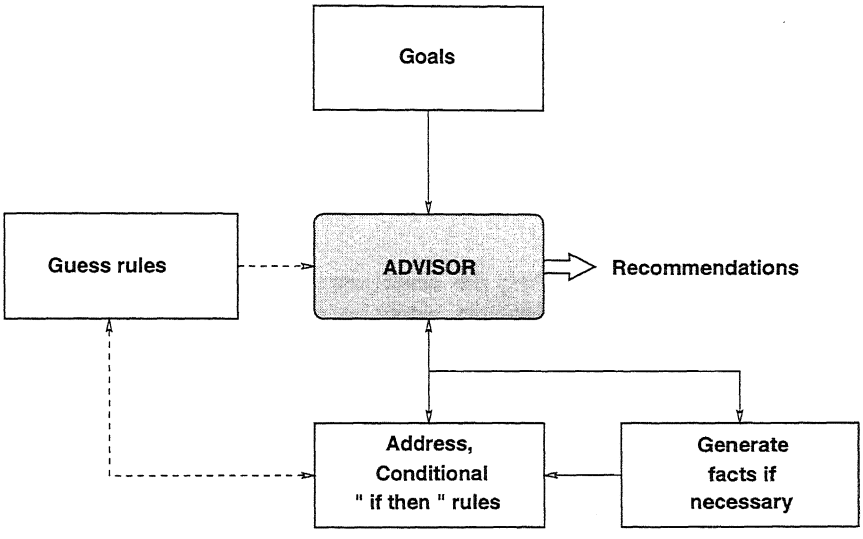


Figure 2.10: Structure of knowledge base.

If knowledge is not found, then our system first checks if such a knowledge can be generated through an interpolation mechanism. For example, suppose the system needs knowledge for a subspace that has the address 23221. Since knowledge for this subspace is not available, the system checks the neighboring subspaces which in the present example were found to be 21321 and 24112. The knowledge stored in each of these subspaces was found to be:

21321 — M0: sNANA M1,M2: NANNN M3,M4 : NNNna M7: nNNSA M8: nNnNA M9: NNnaN Cc: nIlSN

24112 — M0: sNLNA M1,M2: NANNN M3,M4 : NNNna M7: nNNNA M8: nNnSA M9: NNnlN Cc: nIaSN

It is seen that the variation of performances with respect to M1, M2 and M3, M4 are same in the two subspaces. This implies that for address 23221, which is intermediate to the above two subspaces, we can take these variations to be true also. If interpolation approach is not successful, then the circuit simulator is invoked to generate the knowledge. Once it is generated, it is stored in the knowledge base, *i.e.*, knowledge bank gets updated.

Another feature of our system is that it analyzes the knowledge base and merges the neighborhood subspaces which have identical knowledge. In this way the division of performance space into subspaces becomes more realistic. For example, consider the address :

Original addresses

11321 — M0: sNANA M1,M2: NANNN M3,M4 : NNNna M7: nNNSA M8: nNnSA M9: NNnaN Cc: nIlSN
23333 — M0: sNANA M1,M2: NANNN M3,M4 : NNNna M7: nNNSA M8: nNaSA M9: NNnlN Cc: nIlSN

1st, 2nd, 5th Column positions are same

Modified addresses

11321 — M0: sNANA M1,M2: NANNN M3,M4 : NNNna M7: nNNSA M8: nNnSA M9: NNnaN Cc: nIlSN
11331 — M0: sNANA M1,M2: NANNN M3,M4 : NNNna M7: nNNSA M8: nNaSA M9: NNnlN Cc: nIlSN

The digit in the address related to the first specification is now modified from 2 to 1, since it has the same column positions. Similar is the case for second and fifth specifications also.

Our system also tries to determine the tradeoffs between various specifications by noting various specification changes with respect to each other during iterations.

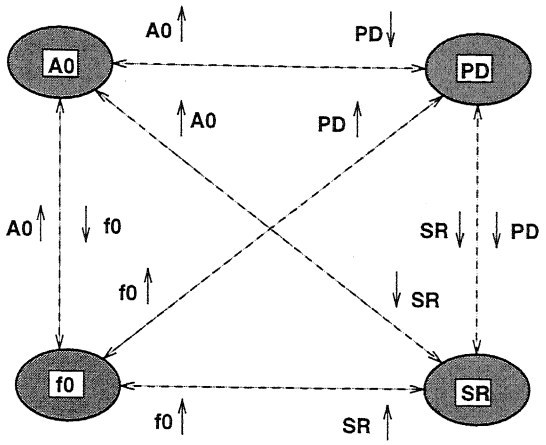


Figure 2.11: Inter–relationships among various performance parameters of an OP–AMP.

Figure 2.11 shows the tradeoffs between OP–AMP performance specifications, which are observed for simple two stage architecture.

The iterations will come to an end when the convergence of the error is less than a predetermined tolerance value. The error is defined as

$$error = \sqrt{\sum_{i=1}^{N}(user_i - simulated_i)^2} \tag{2.2}$$

where

N	:	number of performance specifications.
$user_i$	:	user's i*th* performance specification value.
$simulated_i$	:	i*th* simulated performance specification value.

Design time for a particular architecture decreases with the experience gained by the system.

2.3 The Task Coordinator

LIBRARY
KANPUR
Acc. No. A. 121549

The task coordinator establishes effective communication between the synthesizer, the simulator and the optimizer. After synthesizing the circuit, the synthesizer informs

the task coordinator which then enables the simulator. The interfacing part from the synthesizer to the simulator (SPICE3e2) has been written in *perl* [42] to process output files generated by the simulator. If some problem occurs while simulating, the task coordinator sends messages to the user. It also keeps track of various things like "is error converging for successive iterations" etc. At present we are trying to implement multiple executions of the circuit simulator in parallel [45]. This reduces the simulation time by approximately 'n' times, where n is the number of specifications.

CHAPTER 3

THE PERFORMANCE EVALUATION

From the input performance and the process specifications, our system provides a sized transistor circuit in the form of a SPICE file. At present, the system has the capability of designing bias stages, level shifters, current mirrors, gain stages, differential amplifiers and operational amplifiers using CMOS technology. The optimization using expert system has been implemented only for the simple two stage operational amplifier topology. All the results presented in this chapter are for process parameters listed below :

SPICE model level		=	3	
Process minimum width, length		=	3.0	μm
Supply voltage $V_{DD}, -V_{SS}$		=	3.0	$Volts$
Zero-bias threshold voltage for NMOS	(V_{TNO})	=	0.8	$Volts$
Zero-bias threshold voltage for PMOS	(V_{TPO})	=	-0.8	$Volts$
Transconductance parameter for NMOS	(K_{PN})	=	80.0	$\mu Amp/volt^2$
Transconductance parameter for PMOS	(K_{PP})	=	40.0	$\mu Amp/volt^2$
Bulk threshold parameter	(γ)	=	0.35	$Volt^{1/2}$
Surface potential	(ϕ)	=	0.6	$Volts$
Channel length modulation	(λ)	=	0.03	$Volt^{-1}$
Oxide thickness	(T_{OX})	=	0.1	μm
Surface mobility for NMOS	(μ_{ON})	=	650	$cm^2/volt - sec$
Surface mobility for PMOS	(μ_{OP})	=	350	$cm^2/volt - sec$
Metallurgical junction depth	(X_J)	=	0.3	μm
Lateral diffusion	(L_D)	=	0.25	μm
Zero-bias B-D junction capacitance	(C_{BD})	=	20	fF
Zero-bias B-S junction capacitance	(C_{BS})	=	20	fF
G-S overlap capacitance /meter width	(C_{GSO})	=	10	pF/m
G-D overlap capacitance /metre width	(C_{GDO})	=	10	pF/m
G-B overlap capacitance /meter length	(C_{GBO})	=	200	pF/m
Zero-bias bulk jn. bottom capacitance	(C_J)	=	200	$\mu F/m^2$
Drain ohmic resistance	(R_D)	=	1.0	Ω
Source ohmic resistance	(R_S)	=	1.0	Ω
D-S diff. sheet resistance	(R_{SH})	=	10.0	$\Omega/\square$
Width effect on threshold voltage	(δ)	=	0.5	
Mobility modulation	(θ)	=	0.1	$Volt^{-1}$
Static feedback	(η)	=	0.5	
Saturation field factor	(κ)	=	0.3	
Measurement temparature	(T_{NOM})	=	30	^oC

3.2 The Level Shifters

Two basic topologies as shown in Fig 3.2 have been designed for the level shifter block. If the output voltage is less than the input voltage ($V_{out} < V_{in}$), level shifter 1 is selected, else level shifter 2 is selected. Here, the minimum difference between V_{in} and V_{out} should be equal to the V_{TO} of the transistor. The bias voltage is generated using an appropriate bias stage (shown in dotted lines). Table 3.2 shows the evaluated performance for the level shifter circuits with SPICE.

Parameter	Units	Level Shifter 1		Level Shifter 2	
		Spec	SPICE	Spec	SPICE
V_{in}	Volts	1.0	1.0	-1.0	-1.0
V_{out}	Volts	-1.0	-1.0	1.0	1.0
I_{bias}	$\mu Amps$	50.0	43.2	30.0	25.6

Table 3.2: Evaluation of the level shifters shown in Fig 3.2 with SPICE.

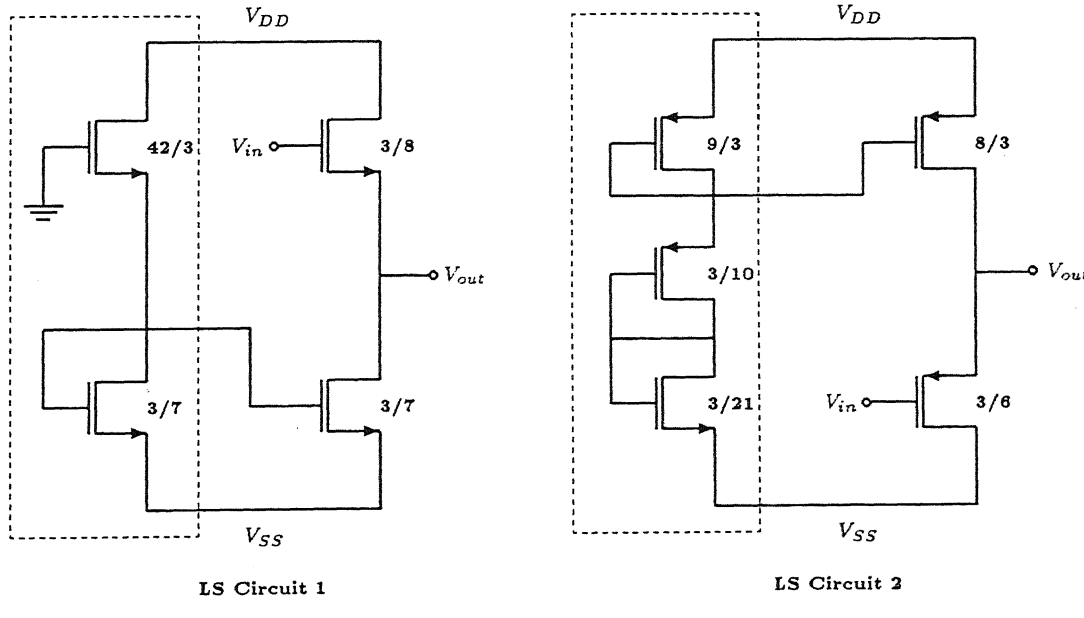


Figure 3.2: Basic topologies of the level shifters.

3.3 The Gain Stages

Only one topology as shown in Fig 3.3 has been designed. Table 3.3 shows the evaluated performance for the gain stage with SPICE.

Parameter	Units	Gain Stage	
		Spec	SPICE
Gain	dB	25.0	24.4
I_{bias}	$\mu Amps$	50.0	46.3

Table 3.3: Evaluation of the gain stage shown in Fig 3.3 with SPICE.

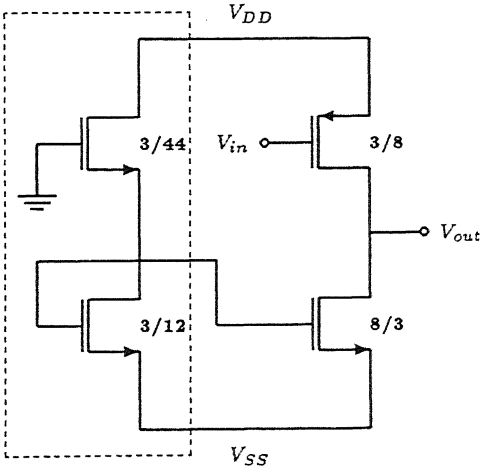


Figure 3.3: A generic gain stage.

3.4 The Current Mirrors

Two topologies as shown in Fig 3.4 have been designed. If the required output resistance is very high($> 100M\Omega$), scheme 2 is selected, else scheme 1 is selected. Table 3.4 shows the evaluted performance for the current mirrors with SPICE.

Parameter	Units	Scheme 1		Scheme 2	
		Spec	SPICE	Spec	SPICE
I_{source}	$\mu Amps$	30.0	25.9	40.0	35.5

Table 3.4: Evaluation of the current mirrors shown in Fig. 3.4 with SPICE.

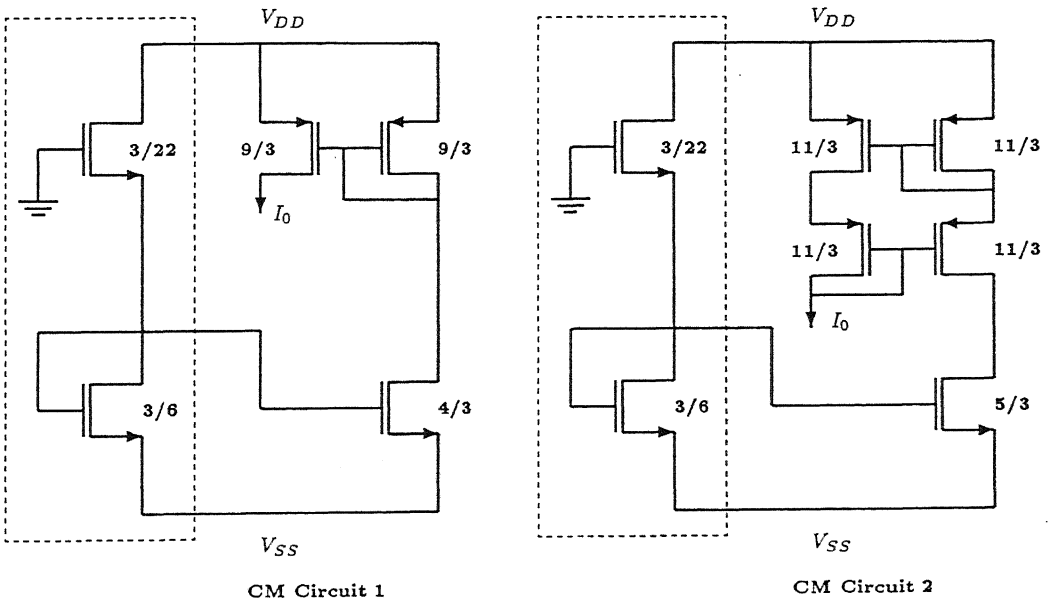


Figure 3.4: Synthesized current mirrors.

3.5 The Differential Amplifiers

The architecture of the differential amplifier is shown in Fig 3.5. Two topologies shown in Fig. 3.6 and Fig. 3.7 have been designed. If the gain required is large ($> 75dB$), the cascode current mirror is selected as the sub-block, since it offers greater output resistance and architecture of scheme 2 will be selected. Bias stages are selected according to the requirement. Table 3.5 shows the evaluated performance for the differential amplifiers with SPICE.

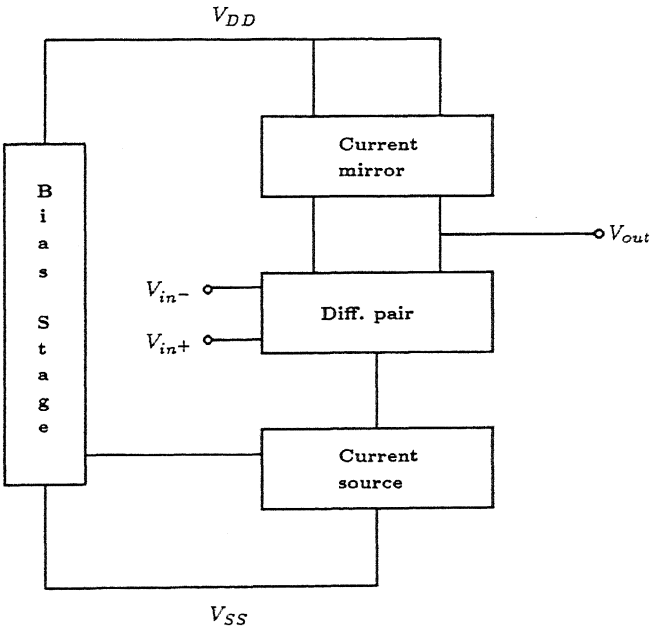


Figure 3.5: Differential amplifier architecture

3.5 The Differential Amplifiers

The architecture of the differential amplifier is shown in Fig 3.5. Two topologies shown in Fig. 3.6 and Fig. 3.7 have been designed. If the gain required is large ($> 75dB$), the cascode current mirror is selected as the sub-block, since it offers greater output resistance and architecture of scheme 2 will be selected. Bias stages are selected according to the requirement. Table 3.5 shows the evaluated performance for the differential amplifiers with SPICE.

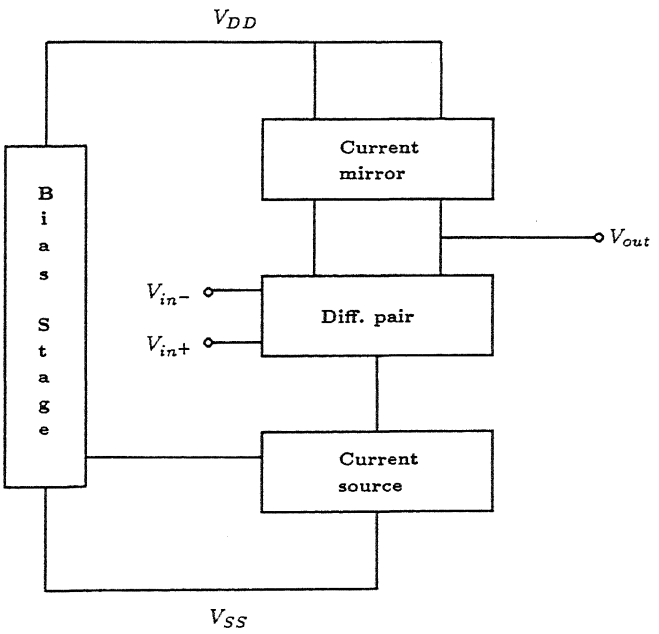


Figure 3.5: Differential amplifier architecture

3.5 The Differential Amplifiers

The architecture of the differential amplifier is shown in Fig 3.5. Two topologies shown in Fig. 3.6 and Fig. 3.7 have been designed. If the gain required is large ($> 75dB$), the cascode current mirror is selected as the sub-block, since it offers greater output resistance and architecture of scheme 2 will be selected. Bias stages are selected according to the requirement. Table 3.5 shows the evaluated performance for the differential amplifiers with SPICE.

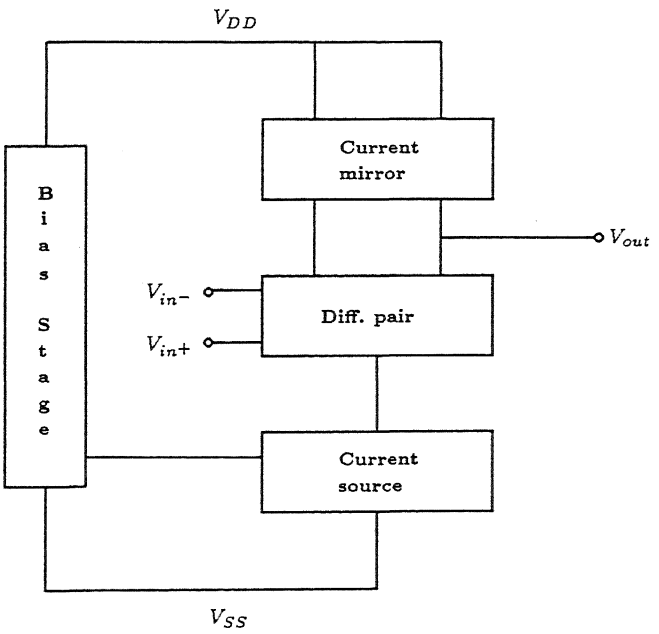


Figure 3.5: Differential amplifier architecture

Parameter	Units	Scheme 1		Scheme 2	
		Spec	SPICE	Spec	SPICE
Gain	dB	40.0	39.7	90.0	89.7
CMRR	dB	45.0	43.7	95.0	96.9
UGF	MHz	3.0	3.2	2.0	2.0
I_0	$\mu Amps$	60.0	59.2	30.0	25.5

Table 3.5: Evaluation of the differential amplifiers shown in Fig. 3.6 & 3.7 with SPICE.

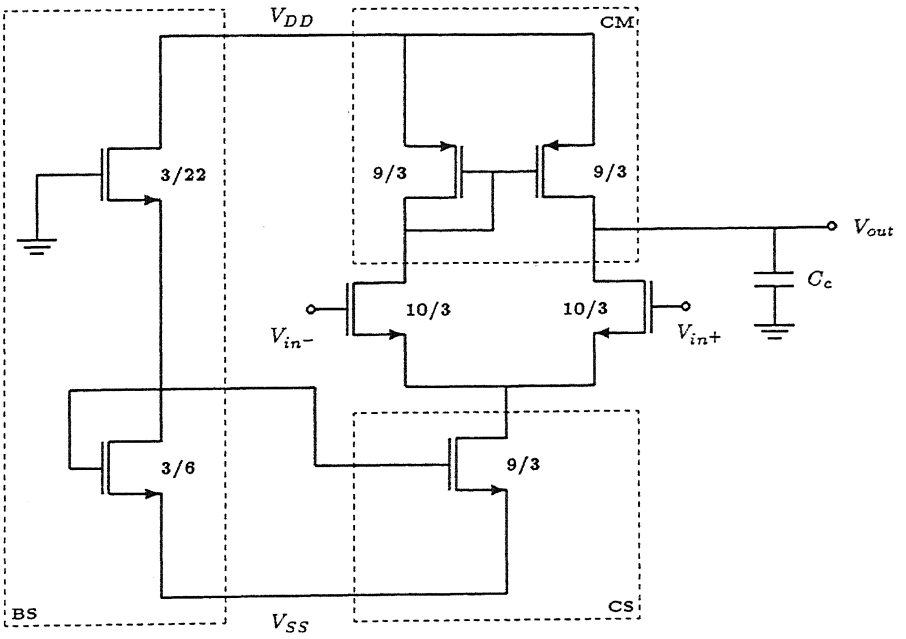


Figure 3.6: Scheme 1 : The schematic of a differential amplifier for moderate gain.

BS : Bias Stage

CS : Current Source

CM : Current Mirror

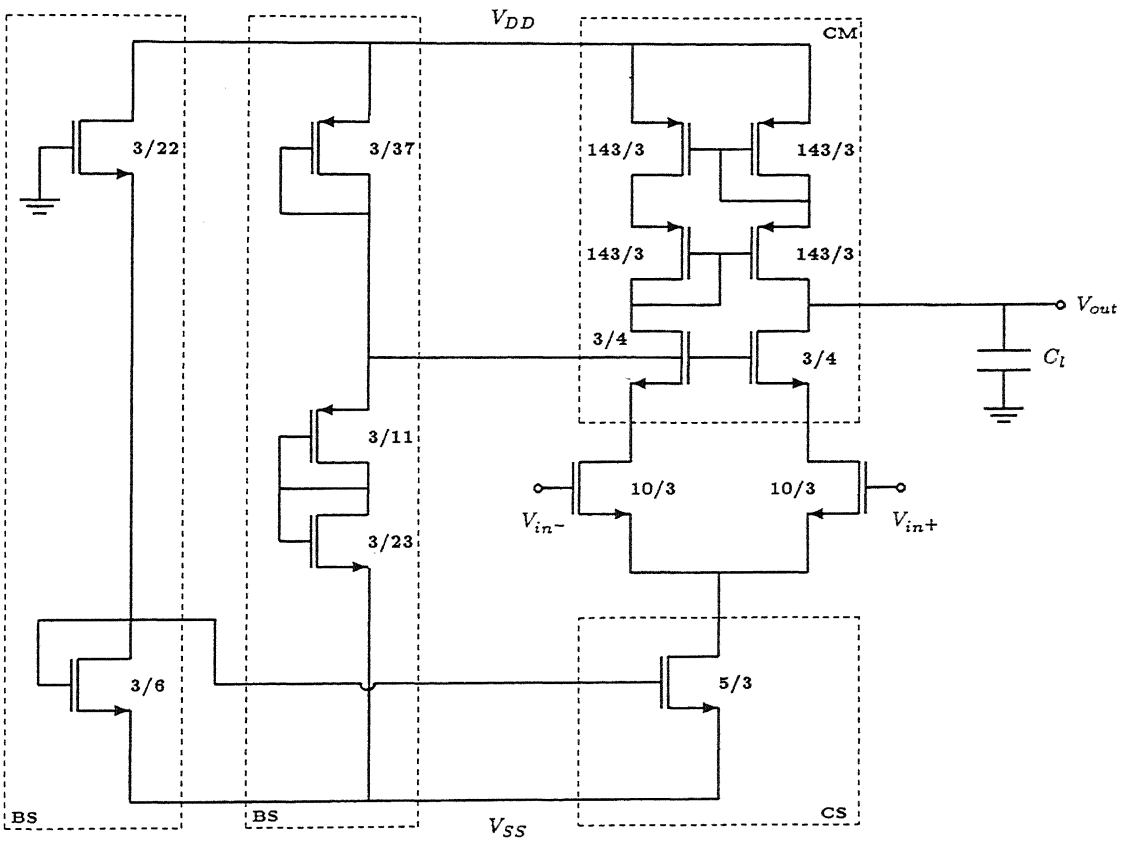


Figure 3.7: Scheme 2: The schematic of a differential amplifier for large gain.

BS : Bias Stage

CS : Current Source

CM : Current Mirror

3.6 The Operational Amplifiers

The system at present has the capability of designing a few commonly used OP-AMP topologies. These topologies have been ranked with respect to performances, such as voltage gain, unity gain frequency, etc. For example,if the gain required is large ($> 85dB$), the cascode current mirror is selected as subblock since it offers more output resistance and architecture of scheme 2 shown in Fig 3.9 will be selected. If the unity gain frequency required is more ($> 5MHz$), then we use folded cascode stage as the subblock. This results in scheme 3 shown in Fig 3.10. Bias stages are selected according to requirement. Table 3.6 shows the evaluated performance for the OP-AMPs with SPICE.

Parameter	Units	Scheme 1		Scheme 2		Scheme 3	
		Spec	SPICE	Spec	SPICE	Spec	SPICE
A_0	dB	> 80.0	79.2	> 110.0	107.3	> 80.0	90.8
UGF	MHz	> 3.0	3.1	> 2.0	2.6	> 6.0	6.7
Slew Rate (+)	v/μ Sec	> 4.0	4.4	> 3.0	3.2	> 3.0	3.3
Slew Rate (-)	v/μ Sec	> - 4.0	- 4.1	> - 3.0	-2.9	> -3.0	-3.3
CMRR	dB	> 85.0	85.1	> 120.0	116.0	> 85.0	131.7
Phase Margin	degrees	> 80.0	115.0	> 80.0	98.0	> 60.0	87.9
Offset Voltage	mVolts	—	0.04	—	0.03	—	-0.02
PSRR (+)@dc	dB	—	85.1	—	88.3	—	131.5
PSRR (-)@dc	dB	—	89.9	—	89.1	—	92.1
PSRR (+)@10kHz	dB	—	79.3	—	77.5	—	116.8
PSRR (-)@10kHz	dB	—	88.0	—	86.3	—	106.9
Power Diss.	mWatts	< 1.0	1.5	< 1.0	1.4	< 1.0	1.9

Table 3.6: Evaluation of the OP-AMPs shown in Figs. 3.8–3.10 with SPICE.

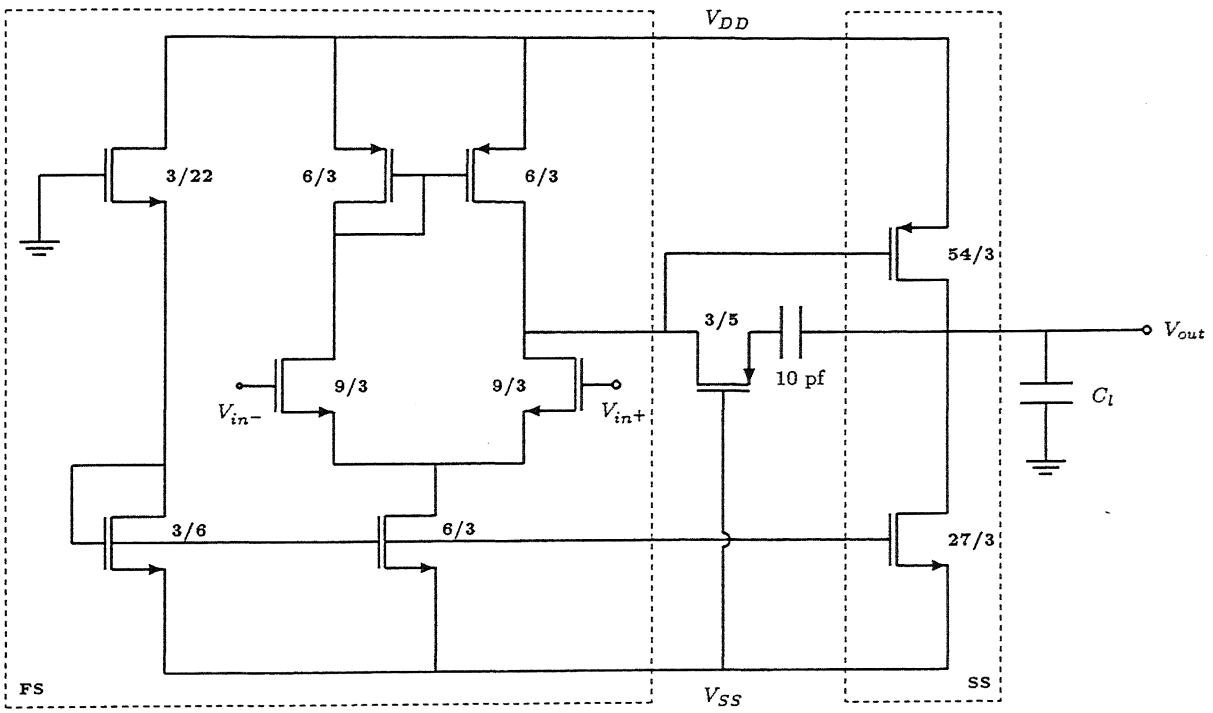


Figure 3.8: Scheme 1: Schematic of an OP-AMP.

FS : First Stage

SS : Second Stage

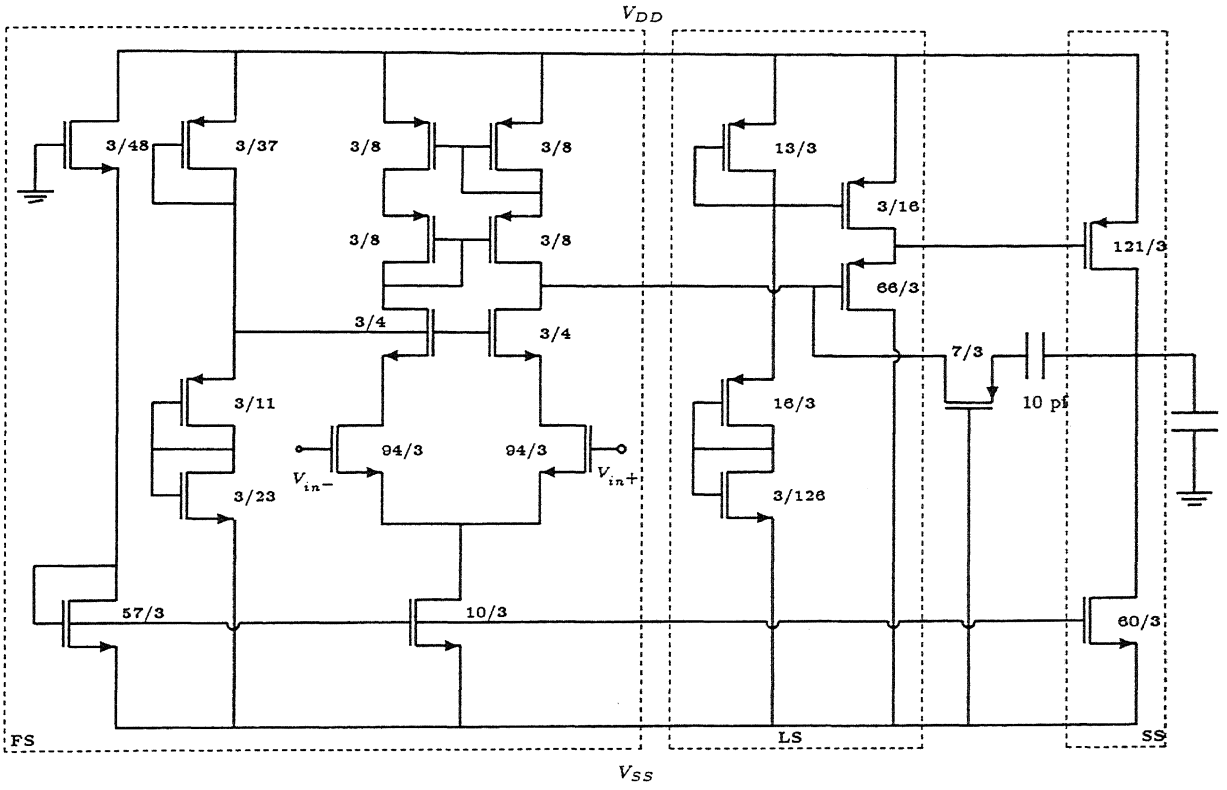


Figure 3.9: Scheme 2: Schematic of an OP-AMP.

FS : First Stage

LS : Level Shifter

SS : Second Stage

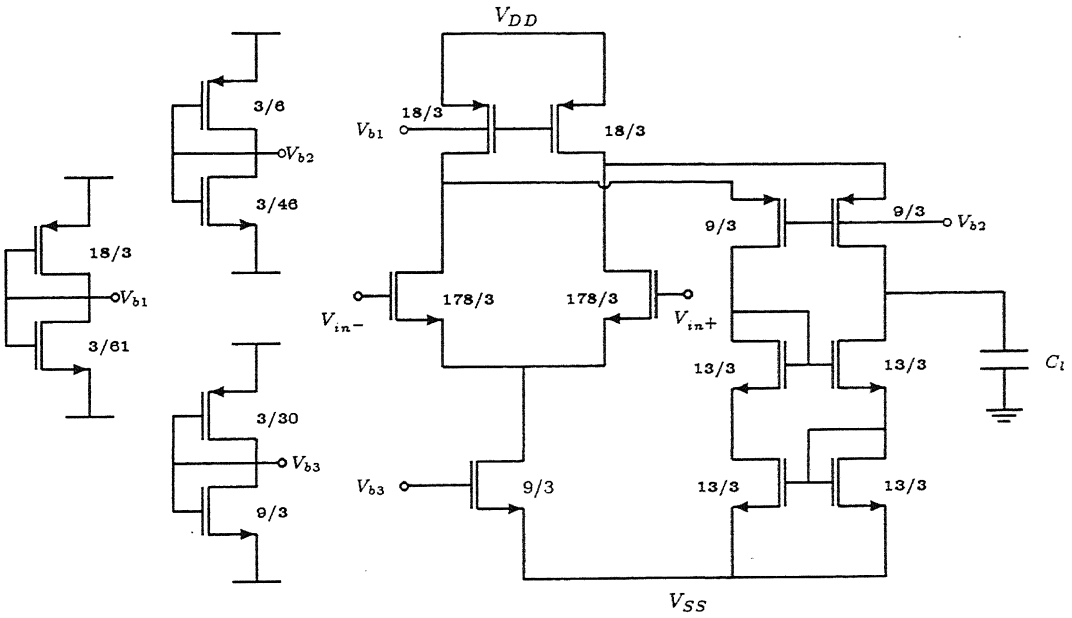


Figure 3.10: Scheme 3: Schematic of an OP-AMP.

After the initial design is completed, it is evaluated using the SPICE simulator. The results are shown in the spec column of Table 3.7.

	OP-AMP	Spec	Initial design	Iter. 1	Iter. 2	Iter. 3	Final design
PERFORMANCES	Avo (dB)	80.0	83.5	84.2	84.9	84.6	82.1
	UGF (MHz)	5.0	6.3	5.9	5.5	5.1	5.1
	SR (Volts/ μSec)	4.0	4.5	4.3	4.1	4.1	4.0
	ϕ_M (Deg.)	75.0	119.0	117.0	116.0	114.0	97.0
	PD (mW)	1.0	1.9	1.7	1.4	1.4	1.5
DEVICE SIZES	M0		6.0/3	5.7/3	5.4/3	5.4/3	5.6/3
	M1, M2		26.0/3	26.0/3	26.0/3	24.7/3	24.7/3
	M3, M4		6.0/3	6.3/3	6.6/3	6.6/3	6.6/3
	M5		3/22	3/22	3/22	3/22	3/22
	M6		3/6	3/6	3/6	3/6	3/6
	M7		70/3	70/3	70/3	70/3	70/3
	M8		35/3	34/3	32/3	30/3	30/3
	M9		3/4	3/4	3/4	3/4	5/3
	Cc		10.0	10.0	10.0	10.0	10.0

Table 3.7: Basic two-stage OP-AMP design example with A_{v0}, f_0, SR, ϕ_M and PD as design specifications.

It is seen that all specs are deviated more than 10% except for A_{vo}. The system therefore enters into the optimization loop in order to correct the circuit. It checks if knowledge for correction of the circuit in its present state exists or not. If not found, then the knowledge is generated using the SPICE simulation and stored into the knowledge base. In the present example, the knowledge was found and is displayed

for the user in the format shown in Table 3.8.

	M0	M1	M3	M7	M8	M9	Cc
Ao	s	N	N	n	n	N	n
UGF	N	A	N	N	N	a	l
SR	L	N	N	N	n	N	a
PM	N	N	n	S	S	a	S
PD	A	N	s	S	S	N	N

n : very small negative *N : very small positive*

s : small negative *S : small positive*

a : average negative *A : average positive*

l : large negative *L : large positive*

x : very large negative *X : very large positive*

Table 3.8: Dependencies of specifications on device sizes.

On the basis of this knowledge, it is seen that the power dissipation (PD) can be decreased by (i) decreasing the size of M_0, (ii) increasing the sizes of M_3,M_4, (iii) decreasing the size of M_7 and (iv) decreasing the size of M_8. It is also seen that the PD has an average dependence on M_0, whereas the changes in the sizes of M_3,M_4, M_7 and M_8 affect it only slightly. Hence, to adjust the PD, the changing the size of the M_0 would be the obvious choice. However, the effects of changing the size of M_0 on other parameters should also be given due considerations. Here, the SR has a large dependence on the size of M_0 and would be adversely affected if we decrease the size of M_0. Fortunately, the SR has an overdesigned value, hence the size of M_0 can be changed, albeit carefully. Take the case of the variations in the sizes of M_3,M_4. Although it affects the PD slightly it has virtually no effect on the other

methodology, it was possible to correct even this very poor design.

	OP-AMP	Spec	Initial design	Iter. 1	Iter. 2	Iter. 3	Final design
PERFORMANCES	Avo (dB)	80.0	74.3	74.7	76.8	77.9	79.3
	UGF (MHz)	2.0	1.7	1.9	2.1	2.1	2.2
	SR (Volts/ μ Sec)	5.0	5.5	5.8	5.0	5.1	5.1
	PSRR_Vdd (dB)	100.0	80.8	80.4	82.4	83.6	84.8
	PSRR_Vss (dB)	100.0	82.4	84.0	87.5	88.8	91.7
DEVICES SIZES	M0		7.5/3	7.9/3	6.7/3	6.7/3	6.7/3
	M1, M2		3.3/3	4.0/3	4.8/3	5.3/3	5.5/3
	M3, M4		8.5/3	8.5/3	8.5/3	9.7/3	13.0/3
	M5		3/22	3/22	3/22	3/22	3/22
	M6		3/6	3/6	3/6	3/6	3/6
	M7		68/3	78/3	93/3	93/3	93/3
	M8		34/3	34/3	34/3	31/3	22/3
	M9		3/4	3/4	3/4	3/4	3/4
	Cc		10.0	10.0	10.0	10.0	10.0

Table 3.9: Basic two-stage OP-AMP design example with A_{v0}, f_0, SR, $PSRR_{V_{DD}}$ and $PSRR_{V_{SS}}$ as design specifications

An expert system has been written to automate the generation of the recommendations, however it could not be linked up because of software difficulties (file system on HP machine where PROLOG is loaded, is not accessible from CONVEX machine where this program is compatible).

CHAPTER 4

CONCLUSIONS

A system that supports a behavioral to a structure synthesis for analog circuits has been implemented. The synthesis procedure starts from a set of input performance and process specifications and generates the circuit meeting these targets adequately. The hierarchical methodology of synthesis involving architecture selection and translation is done using analytical equations and heuristic knowledge. The initial crude design from the synthesizer is refined using an optimizer, which uses an expert system to generate the recommendations for the changes in the circuit. The entire synthesis process is very fast and the system can be used by engineers who may be inexperienced in circuit design.

Source code has been written in ANSI C (the Synthesizer), PERL (the Task Coordinator) and PROLOG (the Advisor) and is compatible with UNIX release 4.0.

SCOPE FOR FUTURE WORK:

- As mentioned earlier, our optimization methodology is intermediate between numerical and traditional rule–based approaches. A detailed comparison between these approaches needs to be done in order to sharpen their relative advantages and disadvantages.

- Our system at present generates only a circuit schematic. The next step is to generate a layout for this circuit. Analog circuits are fairly sensitive to parasitics.

CHAPTER 4

CONCLUSIONS

A system that supports a behavioral to a structure synthesis for analog circuits has been implemented. The synthesis procedure starts from a set of input performance and process specifications and generates the circuit meeting these targets adequately. The hierarchical methodology of synthesis involving architecture selection and translation is done using analytical equations and heuristic knowledge. The initial crude design from the synthesizer is refined using an optimizer, which uses an expert system to generate the recommendations for the changes in the circuit. The entire synthesis process is very fast and the system can be used by engineers who may be inexperienced in circuit design.

Source code has been written in ANSI C (the Synthesizer), PERL (the Task Coordinator) and PROLOG (the Advisor) and is compatible with UNIX release 4.0.

SCOPE FOR FUTURE WORK:

- As mentioned earlier, our optimization methodology is intermediate between numerical and traditional rule-based approaches. A detailed comparison between these approaches needs to be done in order to sharpen their relative advantages and disadvantages.

- Our system at present generates only a circuit schematic. The next step is to generate a layout for this circuit. Analog circuits are fairly sensitive to parasitics.

So a circuit extraction from layout followed by simulation and optimization needs to be incorporated into the system.

- Currently the system supports the synthesis of analog circuits upto the level of macrocell only. It needs to be upgraded for the synthesis of a macromodule or a system.

APPENDIX A

A DESIGN EXAMPLE

Presently, our system is capable of synthesizing the following circuits from user specifications:

(i)	Bias circuits,	(ii)	Level shifters,
(iii)	Gain stages,	(iv)	Current mirrors,
(v)	Differential amplifiers, *and*	(vi)	Operational amplifiers.

The user is first required to choose the circuit type and then enter the performance specifications for the same for a particular process technology. However, the user can change the process technology file. Here, we discuss the synthesizing procedure for an OP-AMP.

A.1 Operational Amplifier Design

The input specifications for the operational amplifier include, apart from the process parameters, the following:

1. Midband gain, A_d
2. Unity Gain Frequency, UGF
3. Common Mode Rejection Ratio, $CMRR$
4. Slew Rate, SR

5. Phase margin, ϕ

6. Power Dissipation, PD and

7. Capacitive Load, C_l.

Before going on to the selection of design styles for the OP-AMP's, we first translate the behavioral specifications to some of the circuit parameters using the following equations:

$$I_0 = (SR) \cdot C_l. \tag{A.1}$$

$$g_{mi} = \omega_0 C_c. \tag{A.2}$$

where I_0 is the bias current of the current source M_0 shown in Fig. A.1 and g_{mi} is the input transconductance of the differential pair M_1, M_2. Now depending on the behavioral specifications and the above parameters, corresponding topology is chosen. If the topology shown in Fig. A.1 is selected, then the following steps are executed:

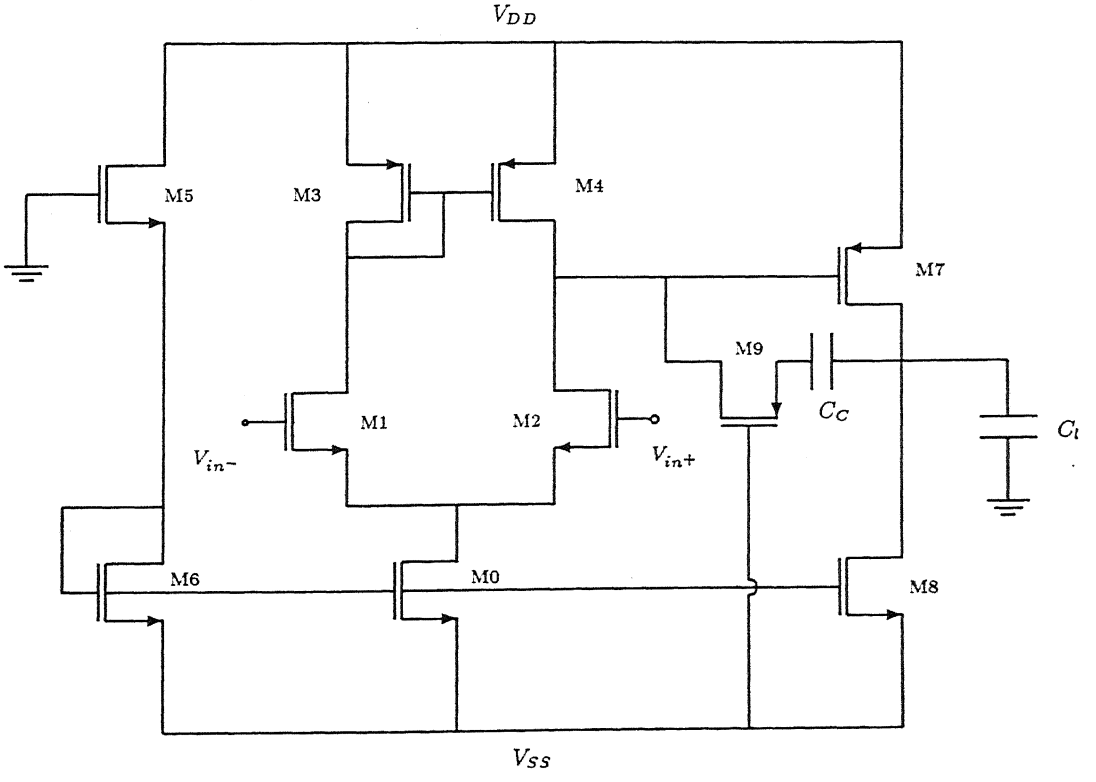


Figure A.1: Architecture of a simple two-stage OP-AMP.

1. First, it would require the translation process for the differential amplifier design. The differential gain required of the differential amplifier A_{d1} can be calculated by using the equation

$$A_{d1} = g_{mi}/(2\lambda \cdot I_0/2) = g_{mi}/(\lambda \cdot I_0). \tag{A.3}$$

The other parameters required are the CMRR, which is reduced by the same amount as A_{d1} is from A_d, the unity gain frequency and the capacitor, which are kept the same and the current value, which is calculated before design style selection of the OP–AMP. Thus a sub-block of OP–AMP is ready.

2. For the design of the gain stage, the gain required A_{d2} is calculated as follows

$$A_{d2}(dB) = A_d(dB) - A_{d1}(dB). \tag{A.4}$$

However since the design of the gain stage is tightly coupled to the OP–AMP specifications, the required value of current, I_{bias} is obtained in two iterations.

(a) The output transconductance is given by

$$g_{mout} = ([tan^{-1}\phi] + 1)\omega_0 C_l. \tag{A.5}$$

(b) The value of I_{bias} is calculated from the equation

$$I_{bias} = g_{mout}/(2\,|V_{GS} - V_T|). \tag{A.6}$$

For the device to be in saturation, we allow the drop $|V_{GS} - V_T|$ to be 0.5 volts.

(c) However to provide for the gain A_{d2}, the transconductance should be greater than

$$g_{mreq} = 2\lambda I_{bias} A_{d2}. \tag{A.7}$$

(d) We choose, therefore, the maximum of g_{mout} and g_{mreq} and assign the value to g_{mout}.

$$g_{mout} = |g_{mout}\,,\, g_{mreq}|_{max}. \tag{A.8}$$

If g_{mreq} is much greater than g_{mout}, then the degradation in the phase margin would be large.

(e) I_{bias} is calculated from this value of g_{mout} using Eqn. A.6.

The translation parameters for the design of the gain stage are now available and the gain stage block is now ready.

3. Design of the compensation network: The compensation network is modeled as a resistor capacitor combination, with the resistor being implemented by the channel resistance of a PMOS device. For the zero, introduced by the compensation network, for canceling the pole s_{p2} introduced by the output PMOS device, the following expression must hold:

$$R_c = 1/|s_{p2}|C_c + 1/g_{m7} \simeq 2/g_{m7}. \tag{A.9}$$

Since the transconductance of the output PMOS device is known, we can determine the value of R_c. The value of R_c is related to the device size by the relation

$$1/R_c = 2k_p(W/L)_9(|V_{SS} - V_{D9}| - |V_T|). \tag{A.10}$$

and since V_{D9} is at the same potential as the drain of the current mirror device, which has already been determined, the aspect ratio of the device $(W/L)_9$ and hence the size can be determined. The compensating capacitor value is taken as approximately the same as the load capacitor.

4. Knowing the values of the bias voltages required, the required bias circuits can be generated.

The OP–AMP design is now complete.

APPENDIX B

EXPERT SYSTEMS

Expert systems(ES) have emerged as one of the most successful Artificial Intelligence(AI) technologies and has reached a significant level of commercialization. Expert systems are intelligent computer programs which store the expertise knowledge and information of human experts. Consider the case of human problem solving approach as shown in Fig. B.1. Human takes a decision from the expertise knowledge and the information available to do an action for solving problems. The expertise knowledge comes from education(training and instruction designed to give knowledge), experience(knowledge acquired from seeing and doing things) and intuition(understanding things immediately, without the need for conscious reasoning). Information comes from facts used in deciding things.

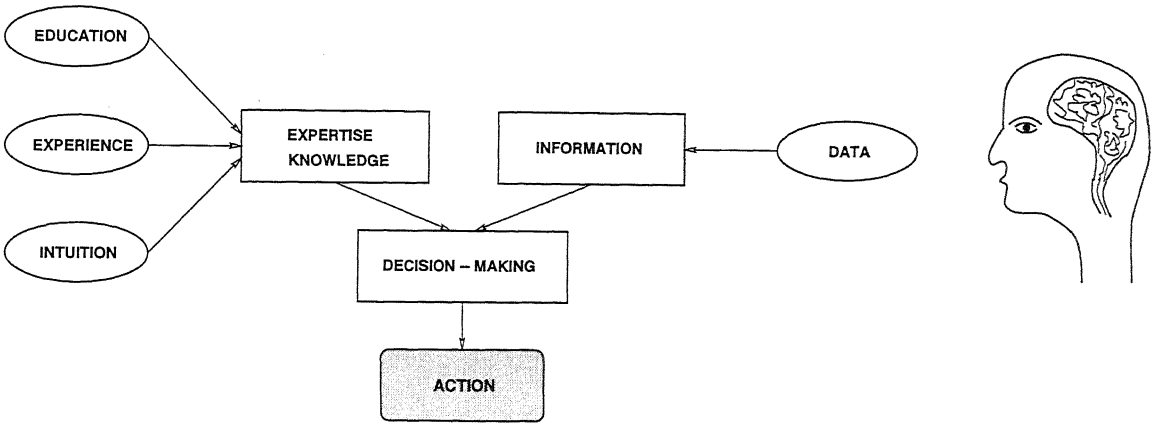


Figure B.1: Human problem solving approach.

The importance for adopting this technology has got the following advantages:

1. The knowledge used for problem solving is transparent. In conventional programs, the knowledge is embedded in the code and is distributed throughout the program, thereby making it extremely difficult to update the knowledge that is used. The transparent nature of the knowledge base allows the user to gain greater insight into the program.

2. An ES can assist and advice users, provide explanations to users at any point of the solution. It can also tell the user why it needs certain information at different stages of problem solving, thereby increasing his confidence in the effectiveness of the system.

3. An expert system can combine the knowledge of many experts. Therefore, theoretically it is possible for the ES to perform at a higher level than any single human expert.

4. Most of the experts do not have time to transfer the knowledge to the juniors with the result that it may be lost when the expert leaves the field. ES enable us to capture the knowledge of co-operating experts and use it forever.

B.1 Rule-Based Systems

First generation expert systems were rule based systems. These are composed of rules, working memory and a rule interpreter. In their simplest form, rules are condition-action pairs:

$$\textbf{IF} < \text{condition} > \textbf{THEN} < \text{condition} >$$

If some condition is true, then some action is performed. Working memory is used to store facts created by rules. The rule interpreter applies a rule when its condition is satisfied by the elements in the working memory. Here is a part of simple rule base:

$$\text{Rule R1} = \begin{cases} IF & A_0 \text{ of OP-AMP to be increased} \\ AND & f_0 \text{ of OP-AMP to be decreased} \\ THEN & \text{decrease aspect ratio } (W/L)_0 \end{cases}$$

$$\text{Rule R2} = \begin{cases} IF & f_0 \text{ of OP-AMP to be increased} \\ AND & \text{Slew rate of OP-AMP to be increased} \\ THEN & \text{increase aspect ratio } (W/L)_0 \end{cases}$$

If rules are independent, then these can be added to a knowledge base(KB) without change in the existing rules.

B.2 Knowledge Acquisition

The construction of KB systems [38] is a complex task. There is a need to acquire the knowledge from a group of recognized experts. The task is difficult and time consuming and is often called the knowledge acquisition bottleneck in an expert system design. Knowledge which is driven from mathematical procedures and system equations is generally known as the "formal knowledge" to distinguish it from the knowledge obtained from experience and observations which is normally termed as the "informal or intuitive knowledge".

B.3 Knowledge Representation

Most of the KB tools [39] represent the knowledge in the form of rules. A useful view of an ES is as a programming methodology that emphasizes the separation of what is true about world (facts) from how to use knowledge to solve problems (rules). In this view, an ES has two components:

(i) Domain knowledge (ii) Problem solving methods

All rule based systems incorporate the following assumptions:

1. An ES incorporates practical knowledge expressed in terms of IF $\cdots$ THEN $\cdots$ rules.
2. An ES grows in skill as its collection of rules is incrementally expanded.
3. An ES can solve a wide range of problems by selecting relevant rules and combining their results in appropriate ways.
4. An ES determines dynamically the best rules to execute.

Some well known ES tools are OPS, INSIGHT, PROLOG, etc.

VAX OPS5 has a lisp like syntax for its production rules. General form of this is:

```
(PRule name
        (Condition - 1)
        (Condition - 2)
              ⋮
        (Condition - m)
⇒       (action - 1)
        (action - 2)
              ⋮
        (action - n)).
```

B.4 Machine Learning

The ability to learn from experience and through instruction is an important attribute of intelligent behavior [41]. Learning can be thought of as extracting deeper insights into

a repeated type of situation, so that one's problem solving expertise is improved with experience. Learning can be defined as change in the system's KB, by its interaction with an external environment. The important area where learning is useful is automated knowledge acquisition in large scale KB systems, such as analog circuit synthesis. Here, it is impossible for us (though all of us are experts at this task) to remember all the performance trends for all values towards device sizes, more accurately. In such problems, one of the useful approaches is to let the system learn the necessary facts and decision rules.

B.5 Implementation

The operational amplifier is one of the most important building block in analog IC design. Here, we address our optimization approach through the design of an operational amplifier.

A typical set of crucial performance specifications of an OP–AMP design includes the open–loop voltage gain A_{vo}, the unity gain frequency f_0, the slew rate SR, the phase margin PM and the power dissipation PD. All these specifications are highly interrelated. It is evident that improvement made on one of the performance specification may very well result in deterioration of the other specifications during optimization. Here, we describe the decision making mechanism for one iteration of our optimization procedure. Consider the case for a particular address of an OP–AMP. Its knowledge in the knowledge base is stored in the form of facts. Expert system implemented in Prolog [43] uses those facts and rules for the decision making. Information is manipulated by way of inference. For example, the rule

IF [the % deviation in PD is more among performance objectives]
THEN [improve PD first]

is implemented as:

Predicates:

fact(spec,parameter,relation)

less_fact(relation,relation)

complement(relation,relation)

Clauses:

```
fact(a0,m0,ns).
fact(a0,m1,pvs).
fact(f0,m1,pa).
    :
fact(pd,cc,pvs).
   change(pd,Parameter) :-
                  fact(pd,Parameter,Value),
                  max_change(pd,Parameter),
                  min_change(a0,Parameter,Value),
                  min_change(f0,Parameter,Value),
                  min_change(sr,Parameter,Value),
                  min_change(pm,Parameter,Value),
                  write(Parameter),
                  complement(Value, Comp_Parameter),
                  write(Comp_Parameter).
   max_change(X,Y) :-
                  fact(X,Y,Z),
                  check(X,Z),
                  !.
   check(Row,X) :-
                  fact(Row,_,Y),
                  less_abs(X,Y),
                  !,
                  fail.
   check(Row,X).
   min_change(Spec,Parameter,Other) :-
                  fact(Spec,Parameter,Value),
                  less_abs(Value,Other).
   less_abs(X,Y) :-
```

```
                    abs(X,AX),
                    abs(Y,AY),
                    less(AX,AY).
   abs(V1,V2) :-
                    name(V1,[pn | rest]),
                    name(V2,rest).
less_fact(vs,s).
less_fact(s,a).
less_fact(a,l).
less_fact(l,vl).
   less(X,Y) :-
                    less_fact(X,Y).
   less(X,Y) :-
                    less_fact(X,Z),
                    less(Z,Y).
complement(pvl,nvs).
complement(pl,ns).
complement(pa,na).
complement(ps,nl).
complement(pvs,nvl).

   complement(X,Y) :-
                    complement(Y,X).
```

Bibliography

[1] G.Gielen, K.Swings, W.Sansen, "An Intelligent Desin System for Analog Integrated Circuits", *Proc. IEEE Euro. Design Automation Conf. (EDAC)*, pp. 169-173, 1990.

[2] W.Nye, D.C.Riley, A.S.Vincentelli, A.L.Tits, "DELIGHT.SPICE: An Optimization-Based System for the Design of Integrated Circuits", *IEEE Trans. on Computer Aided Design*, Vol. 7, No. 4, pp. 501-519, April 1988.

[3] J.C.Lai, J.S.Kueng, H.J.Chen, F.J.Fermandez, "ADOPT: A CAD System for Analog Circuit Design", *Proc. IEEE Custom Integ. Circ. Conf. (CICC)*, pp. 321-324, 1988.

[4] G.Kelson, "Design Automation Techniques for Analog VLSI", *VLSI Design*, pp. 78-82, Jan. 1985.

[5] D.C.Stone, J.E.Schroeder, R.H.Kaplan, A.R.Smith, "Analog CMOS Building Blocks for Custom and Semicustom Applications", *IEEE J. of Solid State Circuits*, Vol.SC-19, No. 1, pp. 55-61, April 1984.

[6] T.Pletersek, J.Trontelj, L.Trontelj, I.Jones, G.Shenton, "High-Performance Designs with CMOS Analog Standard Cells", *IEEE J. of Solid State Circuits*, Vol.SC-21, No. 2, pp. 215-222, April 1986.

[7] P.E.Allen, E.R.Macaluso, S.F.Bily, "AIDE2: An Automated Analog IC Design System", *Proc. IEEE Custom Integ. Circ. Conf. (CICC)*, pp. 498-501, 1985.

[8] M.G.R.Degrauwe, O.Nys, E.Dijkstra, J.Rijmenants, S.Bitz, B.L.A.G.Goffart, E.A.Vittoz, S.Cserveny, C.Meixenberger, G.Vanderstapen, H.J.Oguey, "IDAC: An

Interactive Design Tool for Analog CMOS Circuits", *IEEE J. of Solid State Circuits*, Vol.SC-22, No. 6, pp. 1106-1116, Dec. 1987.

[9] F.EL-Turky, E.E.PERRY, "BLADES: An Artificial Intelligence Approach to Analog Circuit Design", *IEEE Trans. on Computer Aided Design*, Vol. 8, No. 6, pp. 680-692, June 1989.

[10] M.Degrauwe *et al.*, "An Analog Expert Design System", *Proc. IEEE Int. Solid State Circuits Conf. (ISSCC)*, pp. 212-213, 1987.

[11] R.Harjani, R.A.Rutenbar, L.R.Carley, "OASYS: A Framework for Analog Circuit Synthesis", *IEEE Trans. on Computer Aided Design*, Vol. 8, No. 12, pp. 1247-1266, Dec. 1989.

[12] E.Berkcan, F.Yassa, "Towards Mixed Analog/Digital Design Automation: A review", *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 809-815, May 1990.

[13] H.Y.Koh, C.H.Sequin, P.R.Gray, "OPASYN: A Compiler for CMOS Operational Amplifiers", *IEEE Trans. on Computer Aided Design*, Vol. 9, No. 2, pp. 113-125, Feb. 1990.

[14] Y.P.Tsividis, "Design Considerations in Single-Channel MOS Analog integrated Circuits", *IEEE J. of Solid State Circuits*, Vol.SC-13, No. 3, pp. 383-391, June 1978.

[15] P.R.Gray, R.G.Meyer, "MOS Operational Amplifier Design - A Tutorial Overview", *IEEE J. of Solid State Circuits*, Vol.SC-17, No. 6, pp. 969-981, Dec. 1982.

[16] M.G.R.Degrauwe, W.M.C.Sansen, "The Current Efficiency of MOS Transconductance Amplifiers", *IEEE J. of Solid State Circuits*, Vol.SC-19, No. 3, pp. 349-359, June 1984.

[17] R.J.Bowman, D.J.Lane, "A Knowledge-Based System for Analog Integrated Circuit Design", *Proc. IEEE Int. Conf. Computer Aided Design*, 1985.

[18] J.W.Fattaruso, R.G.Meyer, "MOS Analog Function Synthesis", *IEEE J. of Solid State Circuits*, Vol.SC-22, No. 6, pp. 1056-1063, 1987.

[19] R.Harjani, R.A.Rutenber, L.R.Carley, "A Prototype Framework for Knowledge-Based Analog Circuit Synthesis", *Proc. IEEE Design Automation Conf.*, pp. 42-49, 1987.

[20] B.J.Sheu, A.H.Fung, Y.N.Lai, "A Knowledge-Based Approach to Analog IC Design", *IEEE Trans. on Circuits and Systems*, Vol. 35, No. 2, pp. 256-258, Feb. 1988.

[21] L.R.Carley, R.A.Rutenbar, "How to Automate Analog IC Designs", *IEEE Spectrum*, pp. 26-30, Aug. 1988.

[22] J.M.Cohn, D.J.Garrod, R.A.Rutenbar, L.R.Carley, "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing", *IEEE J. of Solid State Circuits*, Vol.SC-26, No. 3, pp. 330-342, Mar. 1991.

[23] Y.P.Tsividis, K.Suyama, "MOSFET Modeling for Analog Circuit CAD: Problems and Prospects", *IEEE J. of Solid State Circuits*, Vol.SC-29, No. 3, pp. 210-216, Mar. 1994.

[24] T.Quarles, "SPICE3e2 Preliminary Report", *University of California, Berkeley*, 1991.

[25] C.Toumazou, C.A.Makris, "Analog IC Design Automation: Part I - Automated Circuit Generation: New Concepts and Methods", *IEEE Trans. on CAD of IC and systems*, Vol. 14, No. 2, pp. 218-238, Feb. 1995.

[26] C.A.Makris, C.Toumazou, "Analog IC Design Automation: Part II - Automated Circuit Correction by Qualitative Reasoning", *IEEE Trans. on CAD of IC and systems*, Vol. 14, No. 2, pp. 239-254, Feb. 1995.

[27] A.H.Fung, B.W.Lee, B.J.Shew, "Self-reconstructing Technique for Expert System based Analog IC Designs", *IEEE Trans. Circuits Syst.*, Vol. 36, No. 2, pp. 318-321, Feb. 1989.

[28] B.A.A.Antao, A.J.Brodersen, "ARCHGEN: Automated Synthesis of Analog Systems", *IEEE Trans. on VLSI Systems*, Vol. 3, No. 2, pp. 231-244, June 1995.

[29] G.Gielen, W.Sansen, "Symbolic Analysis for Automated Design of Analog Integrated Circuits", *Kulwer Academic Publishers*, 1991.

[30] C.Mead, "Analog VLSI and Neural Systems", *Addison-Wesley, MA*, 1989.

[31] C.Mead, M.Ismail, "Analog VLSI Implementation of Neural Systems", *Kulwer Academic Publishers*, 1989.

[32] M.R.Haskard, I.C.May, "Analog VLSI Design: NMOS and CMOS", *Prentice-Hall, NY*, 1988.

[33] M.Ismail, J.Franca, "Introduction to Analog VLSI Design Automation", *Kulwer Academic Publishers*, 1990.

[34] J.H.Huijsing, R.J.V.Plassche, W.Sansen, "Analog Circuit Design", *Kulwer Academic Publishers*, 1993.

[35] R.Gregorian, G.C.Temes, "Analog MOS Integrated Circuits for Signal Processing", *Wiley, New York*, 1986.

[36] P.R.Gray, R.G.Meyer, "Analysis and Design of Analog Integrated Circuits", *John Wiley & Sons, Inc.*, 1993.

[37] V.V.S.Sarma, N.Viswanadham, B.Y.Narayana, B.L.Deekshatulu, "Artificial Intelligence and Expert System Technologies in the Indian Context", *Tata McGraw-Hill Publishing Company Ltd.*, 1991.

[38] S.Ramani, R.Chandrasekar, K.S.R.Anjaneyulu, "Knowledge Based Computer Systems", *Narosa Publishing House*, 1990.

[39] W.Mettrey, "A Comparative Evaluation of Expert System Tools", *IEEE Computer*, Vol. 24, No. 2, pp. 19-31, 1991.

[40] R.Engelmore, "Reading from the AI Magazine", *American Association for Artificial Intelligence, California*, 1988.

[41] J.W.Shavlik, T.G.Dietterich, "Readings in Machine Learning", *Morgan Kaufmann Publishers, Inc.*, 1990.

[42] Larry Wall, Randal L.Schwartz, "Programming Perl", *O'Reilly & Associates, Inc.*, 1990.

[43] W.F.Clocksin, C.S.Mellish, "Programming in Prolog", *Narosa publishing House*, 1993.

[44] S.K.Gupta, "KANSYS: A CAD Tool for Analog Circuit Synthesis", *M-Tech Thesis, Dept. of EE, IIT Kanpur*, Feb. 1995.

[45] D.D.Kumar, "Automatic Parallelization of Sequential Programs for Network Based Computation", *M-Tech Thesis, Dept. of CSE, IIT Kanpur*, To be submitted.